21世纪高等学校规划教材 | 计算机应用



动态网站开发教程

赵景秀　毛书朋　高仲合　编著

清华大学出版社

21 世纪高等学校规划教材·计算机应用

动态网站开发教程

赵景秀　毛书朋　高仲合　编著

清华大学出版社
北　京

内 容 简 介

本书围绕Web标准，结合动态网站开发的实践，详细介绍了用PHP开发动态网站的系统知识。全书共11章，内容涵盖动态网站的概念、XHTML网页结构设计、CSS网页显示控制、JavaScript网页行为操作、PHP语言基础、PHP数据处理、PHP面向对象机制、PHP异常处理、MySQL数据库系统的多种操作方法、Ajax异步通信技术以及XML数据操作等动态网站开发的必备知识。

本书既介绍开发环境的配置，又介绍开发的技术和理论。全书一百二十多个实例，都经过精心测试。这些实例是作者多年来开发实践和课堂教学的结晶，不仅能培养读者的学习兴趣，而且能提高读者的实践动手能力，有的甚至不用修改就可直接用到开发实践中。本书内容丰富，知识全面，讲解详细，操作性强。

本书适合学习动态网站开发的初学者，尤其适合作为高等院校计算机本、专科及相关专业的教材，也可供有一定经验的动态网站开发者参考。

图书在版编目（CIP）数据

动态网站开发教程 / 赵景秀等编著. —北京 ：清华大学出版社，2012.8
（21世纪高等学校规划教材·计算机应用）
ISBN 978-7-302-28835-0

Ⅰ.①动…　Ⅱ.①赵…　Ⅲ.①网站－设计－高等学校－教材　Ⅳ.①TP393.092.1

中国版本图书馆CIP数据核字（2012）第102453号

责任编辑：郑寅堃　赵晓宁
封面设计：焦丽丽
责任校对：
责任印制：

出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn，　http://www.wqbook.com
　　地　　址：北京清华大学学研大厦A座　　　**邮　　编：**100084
　　社 总 机：010-62770175　　　**邮　　购：**010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
　　课件下载：http://www.tup.com.cn,010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张：**20.25　　**字　数：**495千字
版　　次：2012年8月第1版　　**印　次：**2012年8月第1次印刷
印　　数：1～ 0000
定　　价：00.00元

产品编号：043068-01

编审委员会成员

（按地区排序）

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施"高等学校本科教学质量与教学改革工程（简称'质量工程'）"，通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部"质量工程"的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程（群）加以规划、整理和总结，更新教学内容、改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合"质量工程"的实施，满足各高校教学质量和教学改革的需要。

为了深入贯彻落实教育部《关于加强高等学校本科教学工作，提高教学质量的若干意见》精神，紧密配合教育部已经启动的"高等学校教学质量与教学改革工程精品课程建设工作"，在有关专家、教授的倡议和有关部门的大力支持下，我们组织并成立了"清华大学出版社教材编审委员会"（以下简称"编委会"），旨在配合教育部制定精品课程教材的出版规划，讨论并实施精品课程教材的编写与出版工作。"编委会"成员皆来自全国各类高等学校教学与科研第一线的骨干教师，其中许多教师为各校相关院、系主管教学的院长或系主任。

按照教育部的要求，"编委会"一致认为，精品课程的建设工作从开始就要坚持高标准、严要求，处于一个比较高的起点上；精品课程教材应该能够反映各高校教学改革与课程建设的需要，要有特色风格、有创新性（新体系、新内容、新手段、新思路，教材的内容体系有较高的科学创新、技术创新和理念创新的含量）、先进性（对原有的学科体系有实质性的改革和发展，顺应并符合 21 世纪教学发展的规律，代表并引领课程发展的趋势和方向）、示范性（教材所体现的课程体系具有较广泛的辐射性和示范性）和一定的前瞻性。教材由个人申报或各校推荐（通过所在高校的"编委会"成员推荐），经"编委会"认真评审，最后由清华大学出版社审定出版。

目前，针对计算机类和电子信息类相关专业成立了两个"编委会"，即"清华大学出版社计算机教材编审委员会"和"清华大学出版社电子信息教材编审委员会"。推出的特色

精品教材包括：

（1）21 世纪高等学校规划教材 • 计算机应用——高等学校各类专业，特别是非计算机专业的计算机应用类教材。

（2）21 世纪高等学校规划教材 • 计算机科学与技术——高等学校计算机相关专业的教材。

（3）21 世纪高等学校规划教材 • 电子信息——高等学校电子信息相关专业的教材。

（4）21 世纪高等学校规划教材 • 软件工程——高等学校软件工程相关专业的教材。

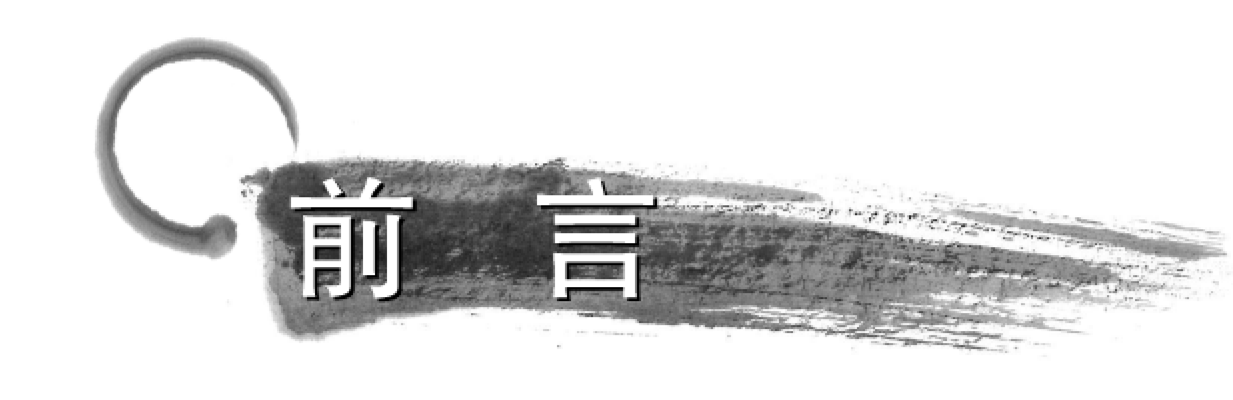
（5）21 世纪高等学校规划教材 • 信息管理与信息系统。

（6）21 世纪高等学校规划教材 • 财经管理与计算机应用。

（7）21 世纪高等学校规划教材 • 电子商务。

清华大学出版社经过二十多年的努力，在教材尤其是计算机和电子信息类专业教材出版方面树立了权威品牌，为我国的高等教育事业做出了重要贡献。清华版教材形成了技术准确、内容严谨的独特风格，这种风格将延续并反映在特色精品教材的建设中。

清华大学出版社教材编审委员会
联系人：　魏江江
E-mail:weijj@tup.tsinghua.edu.cn



计算机相关专业中众多成功者的第一个项目往往是给客户建立一个网站。的确，在当今的信息时代，网站越来越成为单位、组织乃至个人不可或缺的组成部分。政府通过网站实施一定的社会管理功能；企业通过网站宣传产品，组织企业资源；人们通过网站传播自己的影响；网站也使人们获得了方方面面的信息，丰富并改变着人们的生活，网站的开发有广泛的社会需求。在各种开发网站的技术中，PHP 以其简单易用、功能强大而备受欢迎。尤其是 PHP 5 版本的发布，标志着 PHP 面向对象的功能已逐渐成熟，它几乎可以实现.NET 以及 J2EE 等架构所能提供的所有功能，并且在某些方面可能会更优越。与之搭配完美的数据库系统 MySQL 也从此版本完善了存储过程、事务处理等功能，PHP+MySQL 已完全可以放心地用于部署大型系统，PHP 与 MySQL 配合用于各类网站开发已被广泛采用。目前，关于 PHP 开发应用的书已经很多，但是适合作为高等院校教材的很少，为了满足对 PHP 教材的需求，我们在多年开发的基础上，不断总结教学经验，围绕 Web 标准，编写了本书。

读者对象

本书适合于从事动态网站开发的初中级人员。根据多年的教学体会和实际开发经验，我们慎重地安排了本书的内容。从动态网站的基本概念入手，到有一定深度的 Ajax 和 XML 操作技术；从前台页面的设计到后台数据库的构建，本书为读者从事动态网站开发提供了基础而又全面的内容，提供了大量从实际开发中提炼出来的实用技术，读者甚至不加修改就可以用于自己的开发项目中。通过学习本书，不但能掌握 PHP 动态网站开发的基本步骤，还能培养学以致用的专业素养。

本书结构

本书分为 5 个部分。

第 1 部分，讲述动态网站的基本知识，即第 1 章的内容。这部分是学习动态网站开发必备的基础知识。该部分区别静态网站和动态网站的主要不同，明确了从事动态网站开发需要掌握的基本知识，介绍了动态网站开发的一般步骤。

第 2 部分，讲述动态网站客户端的开发技术，包括第 2～第 4 章。第 2 章讲述 XHTML，选择了最常用的标记介绍它们的属性和用法，这部分体现的是网页的结构化标准。第 3 章讲述 CSS+DIV，介绍了网页的布局和控制方法，是网页的表现标准部分。网页中的行为标准用 JavaScript 完成，即第 4 章的内容，讲述了 JavaScript 的基本语法、浏览器模型和事件处理。这部分以 Web 标准为目标组织内容，讲解上有详有略。

第 3 部分，讲述动态网站服务端的开发技术，包括第 5～第 8 章。第 5 章讲述 PHP 的开发的环境配置，第 6 章讲述 PHP 5 的基本语法，第 7 章讲述 PHP 对数据的处理，第 8 章

讲述 PHP 的面向对象编程模型。这部分内容完整，讲解详细，举例丰富。

第 4 部分，数据库技术，主要讲述 MySQL 数据库的应用，包括第 9 和第 10 章。第 9 章讲述 MySQL 数据库的安装与配置，提供了 MySQL 数据库管理的常用方法，第 10 章详细介绍对 MySQL 数据库的操作方法，着重讲解了 MySQL 和 PDO 的用法，尤其增加了对 MySQL 数据库新增功能如存储过程、事物处理等的高级特性讲解。

第 5 部分，即第 11 章内容。XML 是 Web 设计的发展趋势，具有 4 大特点：优良的数据存储格式、可扩展性、高度结构化以及方便的网络传输，是新一代网页结构的标准。Ajax 以 PHP 和 MySQL 为后台，是提高用户体验的重要技术。第 11 章对这两种技术的基本应用做了介绍。

本书第 1 章由赵景秀和毛书朋共同执笔，第 3、4、6、10、11 章由赵景秀完成，第 7～9 章由毛书朋完成，第 2、5 章由高仲合完成，颜明阳完成了本书大部分的程序代码。本书的所有例子程序全都经过测试，读者可放心使用。

由于作者水平有限，对书中不足之处，欢迎广大读者和同行指正。

联系作者

欢迎读者通过电子邮箱 jingxiuzhao@126.com 与笔者取得联系。本书的 PPT 课件和书中程序源代码可到清华大学出版社网站上下载。

致谢

感谢曲阜师范大学计算机科学学院的领导、同事们给予的可贵的支持和帮助，计算机科学与技术系的同学对本书的内容提出了中肯的建议，网上许多资料丰富了作者的思路，在此向他们一并表示衷心地感谢，并以此书向他们表达我们的敬意。

最后感谢家人对我的理解和支持。

编　者

2012 年 4 月


目 录

第1章 动态网站开发概论

本章重点

- 静态网站和动态网站的概念和区别;
- 网站开发的基本步骤;
- 网站测试的内容和方法。

互联网的发展改变了世界，给人们的生活、工作带来了前所未有的便利。在网络时代，网站成了政府、组织、单位乃至个人不可或缺的组成部分。电子政务的推进，使得政府网站逐步成为政务公开的重要窗口和建设服务型政府的重要平台。通过网站，企业无须建立自己的分支机构或派遣业务人员就可将业务拓展到全国乃至全球，大大提高企业内部、生产者和用户联络沟通的效率。当企业逐渐基于网站开展业务时，企业的业务流程、组织机构、人员素质也将随之改变、优化，管理水平提高，竞争力增强。网站建设成了现代信息社会的一项重要内容。

1.1 什么是动态网站

网站有动静之分。有人以为只要网站上有动画就是动态网站了，其实不然，本节先来弄清这两类不同技术的概念。

1.1.1 静态网站

在网站设计中，纯粹 HTML 格式的网页通常被称为“静态网页”，也就是以 htm、html、shtml、xml 等为后缀的文件。早期的网站一般都是由静态网页组成的，网站就是网页的集合。静态网站是指全部由 HTML 代码格式的静态页面组成的网站，所有的内容包含在网页文件中，网页的内容是固定的，不论谁访问都是不变的。在静态网页中含有一定格式的文件后也可以出现各种视觉动态效果，如 GIF 动画、Flash 动画、滚动字幕等。当用户访问静态网站时，服务器不经过任何运算，直接找到用户请求的页面并输出给用户，所以访问起来速度较快。

静态网站不能直接对网页内容进行修改，维护起来非常烦琐。静态网站没有数据库支

持，只能简单地展示新闻和产品，实现不了会员注册、在线留言等与用户交互的功能。如果网站内容非常多，采用静态网站制作是非常累的，每个页面都要单独制作，其维护更是令人崩溃。所以静态网站只适用于非常简单的网站。

1.1.2 动态网站

动态网站并不是指具有动画功能的网站，而是指能和用户进行交互的网站。动态网站中的网页是动态网页，动态网页文件运行在 Web 服务器上。当用户访问动态网站时，运行动态网站的服务器通过执行动态网页文件，将用户请求的网页计算出来再发送给用户的浏览器展示给用户。动态网页文件通常是用运行于服务器端的脚本语言建立的，页面文件名常以 asp、php、jsp 等为后缀，本书使用 PHP 脚本语言。用户请求某个页面时，服务器首先会对页面中的 PHP 命令进行处理，然后再把处理结果连同 HTML 内容一起传送到用户的浏览器。运行动态网页时，不仅要运行脚本文件，还常常访问数据库。动态网站一般以数据库技术为基础，可以大大降低网站维护的工作量，动态网站可以实现更多的功能，如用户注册、用户登录、在线调查、用户管理、订单管理等。动态网站都有一个后台，网站管理员可以从这里添加、删除、更新内容，网站管理十分方便。

1.1.3 网站开发需要掌握的知识

网站开发是个系统工程，涉及的知识较为繁杂。作为初学者，至少需要掌握以下几个方面的内容：

（1）XHTML。所有的网页无论是动态网页还是静态网页，都是最终形成 XHTML 语言的文件并为浏览器所解释的，所以，学习网站首先必须熟悉 XHTML 语言。

（2）CSS。样式表的出现简化了 XHTML 语言并且减轻了 XHTML 对表现形式的责任，使设计者和用户都可以控制文档的表现形式，包括字体信息、对齐方式、颜色等。

XHTML 和 CSS 一起构成网站的客户端基石，是网站的展示层。

（3）一门编写动态网页的脚本语言，如 PHP、Python 等。动态网页技术和支持它们的底层技术一起也被称为中间件，构成了网络应用的中间层。由于这个中间层包含了用于分析用户不同请求，并对不同请求做出不同响应的业务逻辑代码，因此，这个层又被称为业务逻辑层。

（4）数据库操作语言 SQL。位于业务逻辑层之下的是数据服务层，用户请求所要获取的数据从这里被“筛选”出来，并经过业务逻辑层被发送到客户端，被包含在 XHTML 文件中。

（5）XML。XML 已经成为因特网间标准的数据交换格式，也正在成为数据存储的标准格式。

展示层、业务逻辑层、数据服务层构成了当前万维网应用程序的基础架构，也被统称为网站的“三层架构”。

（6）JavaScript。网页要在有限的页面空间展示更多的内容，增加客户的体验，进而使网站更加有动感、有魅力，吸引更多的浏览者，这些工作的完成都要使用 JavaScript。

（7）了解 Web 服务器。动态网站是运行在服务器上的，现在流行的 Web 服务器如 IIS、Apache 等，进行网站开发需要熟练地配置和操作这些服务器。

（8）掌握几个常用数据库系统的配置与操作，如 MySQL、SQL Server 和 Oracle 等。动态网站的运行离不开数据库的支持，要熟练掌握对数据库的管理方法。

（9）某个好用的开发工具/环境，如 Dreamweave，会使编辑代码事半功倍。

（10）网络安全基本知识。写代码时要注意是否存在溢出和注入漏洞，从代码层开始构筑健壮的网站。

1.2　网站建设的一般步骤

动态网站开发是一个系统工程，有特定的工作流程，无论网站大小，只有遵循这个步骤，按步就班地一步步来，才能完成一个让客户满意的网站。

1.2.1　明确客户需求

接到客户开发网站的订单后，就开始了和客户的交流。开发软件一定要知道：决定开发是否成功的第一要素不是技术多么先进和成熟，而是与客户的交流是否充分、完备。人们有个很有意思的特点，就是绝大多数人尽管喋喋不休地说了一大通，滔滔不绝的话语中以废话居多，用语言表达出来的和他（她）内心想要表达的往往有很大差距。这也难怪，学习过数理逻辑的人还是太少了，大多数人的思维是缺少逻辑、模糊不清的——只是程度上的差异而已。所以客户大多没有能力明确表达自己的需求，开发人员需要耐心反复地同客户交流，尽力弄清客户的真实意图，然后将自己的理解反馈给客户，让客户确认。人与人之间的交流都是使用自然语言表达的。由于人们在各自表达方式、文化背景、生活习惯等多方面的差异存在，使得自然语言的表达对普通人来说，是很容易产生二义性的，相同的描述在不同的人那里的理解常常是不完全一样的，有时甚至是大相径庭的。只要稍微留意，就会发现这种现象生活中比比皆是，于是人们有了各种各样的误解乃至纠纷。所以要想保证开发成功，避免纠纷，务必要同客户充分交流。完备交流是说，从和客户第一次见面，到网站交付给客户后，永远都要充分交流。不这样做的最常见结果就是，当你兴冲冲地将产品展示给客户时，客户却说，我的意思不是这样，我的意思是……，于是烦恼甚至崩溃就不期而至了。记住这点吧，除非想把事情搞糟。通过和客户的初步交流，首先获得客户信任并明确几个问题：

（1）客户所从事的行业。

（2）客户建立网站的目的。

（3）客户在网站上要解决的问题。

（4）客户的网络环境。

（5）让客户成为开发小组的一员。

（6）客户的支付能力。

1.2.2　进行网站需求分析

在前面的基础上，往下要进行的就该是对要开发的网站进行需求分析了。这一阶段，

通过和客户交流，正确引导客户能够将自己的实际需求用较为适当的技术语言进行表达（或由相关技术人员帮助表达）以明确项目目的的过程。这个过程的结果是要得出经客户确认的要建立的网站基本功能和模块。

1. 客户需求分析

需求分析过程中，需要客户的充分配合，有时甚至可能需要对客户方的调查对象进行必要的培训。着重要了解的内容主要如下：

- 网站当前的功能需求；
- 客户对网站的性能要求；
- 建站方式，选择独立主机还是虚拟主机；
- 现有的网络基础；
- 网站的可靠性要求；
- 项目完成时间及进度；
- 页面特效要求；
- 验收方式及依据。

2. 用户需求分析

除此以外，还有一个需求：用户的需求即使用网站者的需求。注意这里的客户和用户是两个不同的概念。用户需求需要和客户一起分析以下内容：

（1）网站是给谁用的？每个网站都应有自己的服务目标，如物流信息网站的客户会有物流公司、搬家公司、车主、货主等，明确了对象，才能洞察消费者需求，满足消费者及合作伙伴全方位要求，才能将产品及服务的各项优势充分地进行网络传播，吸引、稳定目标客户群，提高网站服务的附加值。

（2）网站用户需要什么样的内容？大多数的用户上网都是有目的、有针对性地在寻求某些知识或能够帮助自己的信息。分析用户的真正需求是什么，以便在栏目设计时有的放矢，提供用户需要的服务，进而留住用户、赢得更多的用户。

（3）用户会以怎样的方式登录网站？比如，用户会用什么样的浏览器？用户的屏幕是宽屏还是普通屏？国内用户还是国外用户？考虑不考虑手机用户?这些方面的分析有助于指导网页的设计。

（4）用户通常需要哪些资源和帮助？网站不仅要提供用户需要的信息，还要提供相关的信息资源，这些资源既可以丰富网站内容，又可使用户尽可能方便、快捷地获取信息。

3. 需求分析报告

调查分析结束以后，需要编写需求分析报告，报告包括如下的内容：

（1）情况说明：网站项目的名称；用户单位；参与需求调查的人员；需求分析工作的历程。

（2）需求说明：用户的基本情况；用户的主要业务；用户信息化建设现状；网站当前和将来潜在的功能需求、性能需求、可靠性需求；网站运行环境选择；项目完成时间及进度；页面特效要求；验收方式及依据。

1.2.3 进行系统设计

明确了客户需求并经客户确认后，就可进行网站的系统设计了。系统设计包括以下几

个方面。

1. 确定网站类型

成千上万的网站，大体可分为政府门户信息网站、行业、协会信息门户网站、电子商务网站、企事业门户信息网站等。一旦确定了网站类型，就可到同类型网站上调研，通过分析对比，进行自己的设计。

2. 确定网站风格

网站风格是指网站的整体形象给浏览者的综合感受，包括网站的标志，使用的色彩、字体，版面布局，交互性等方面的内容共同形成网站的整体风格，这是网站设计中最见功夫的一项。

3. 设计功能模块

网站是个系统，按照要实现的不同功能，常常划分为不同的子系统。典型的网站大多包含有新闻发布系统、广告发布系统、在线留言系统、在线支付系统、会员管理系统、自主建站系统等。每个子系统就是一个功能模块。模块的划分，在功能上要尽可能单一和明确化，模块间的联系应尽可能少，也就是所说的“高内聚低耦合”。对于必需的联系都应该有明确的说明，模块应足够小，以方便单个模块的调试。例如，一个可能的多功能多模块广告发布子系统的设计如下：

（1）预留与其他物流企业或高级会员企业醒目链接接口。

（2）可以方便和快速地在首页突出重大活动。

（3）重大节日做特殊的宣传。

（4）方便网站美工的更新。

（5）支持多种媒体文件，如 jpg、gif、bmp、png、swf 等，并可以添加链接、自定义高度等。

（6）与动态多模块新闻发布子系统建立广告池，提供广告的宣传力度。

（7）提供单选和多选两种广告模式。

（8）提供独显和分割两种自由尺寸设计模式。

其拓扑及管理流程如图 1-1 所示。

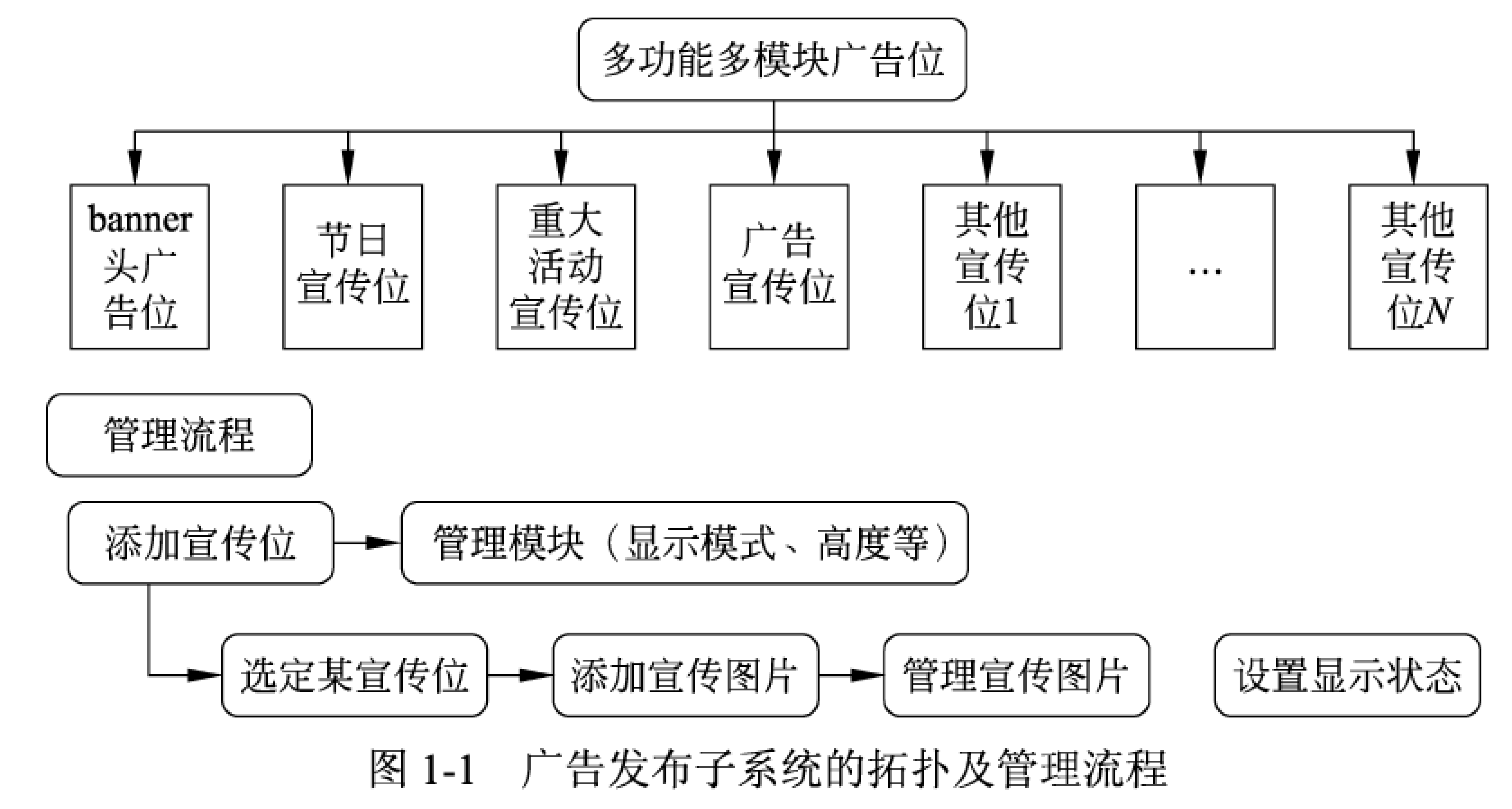


图 1-1　广告发布子系统的拓扑及管理流程

4. 设计好数据库

数据库是动态网站的核心，数据库设计主要是指设计数据表，也就是遵守 3NF 标准

设计数据表。例如，某个表只包括其本身基本的属性，当不是它们本身所具有的属性时就要进行分解。实际上，满足 3NF 的设计往往不是最好的设计，为了提高数据库的运行效率，常常需要适当增加数据冗余，降低范式标准。尽量采用视图，减少应用程序对数据表的直接访问，增强数据的安全性。

5. 确定目录结构

确定网站的目录是为了方便对网站文件的管理。尽量每个栏目建立一个目录，目录的层次尽量别超过 3 层，目录名不用中文，以保证构造正确的网址。管理图像时，在每个主栏目目录下都建立独立的 images 目录，根目录下的 images 目录只是用来放首页和一些次要栏目的图片。数据库务必单独建立目录，最好让用户不能直接访问，不使用用户能猜到内容和能下载的目录名。

6. 设计主页栏目

主页是网站的门面，是用户访问网站的第一个入口，网页布局结构、风格色彩都会给用户留下强烈的“第一印象”，直接冲击用户的视觉感受。主页应尽量主题鲜明，导航醒目易用，内容优化。必要时可请专业美术人员帮助设计。

系统设计完成后，要向用户提交开发设计方案，等用户确认后再实施开发。

1.2.4 上传测试

网站开发完成后，要发布到 Web 服务器上，进行全面测试。测试包括功能测试和性能测试两个方面。

1. 功能测试

通常要进行的测试包括以下 3 个方面。

（1）链接测试：测试所链接的页面是否存在，所有链接是否能正确链接，消除孤立的页面。

（2）表单测试：测试表单提交操作的完整性，以校验提交给服务器的信息的正确性。表单提交应当模拟用户提交，验证是否完成功能，如注册信息，要测试这些程序，需要验证服务器能正确保存这些数据，而且后台运行的程序能正确解释和使用这些信息。还有数据正确性验证，异常处理等，如果表单只能接受指定的某些值，则也要进行测试。测试中要保证每种类型都有两个以上的典型数值的输入，以确保测试输入的全面性。

（3）数据库测试：测试能否正确连接数据库，对数据表的访问是否正确，观察有无输出错误。

2. 性能测试

性能测试主要包括以下 3 个方面。

（1）连接速度测试：连接速度与用户的上网方式有关。一般来说，如果网站响应时间超过 5 秒钟，用户就很可能会失去耐心而离开。连接速度太慢，还可能引起数据丢失。当发现速度太慢时要分析是网络原因还是程序原因，如是程序原因就要修改算法，重写代码。

（2）负载测试：通过模拟大批量用户的并发请求，给网站施加较大的负载，检测整个系统是否在需求范围内能正常工作。例如，网站能允许多少个用户同时在线？如果超过了这个数量，会出现什么现象？负载测试实现的前提是要先准备巨大的数据量，如上百兆的

文件、上万的用户等。负载测试不以使系统崩溃为目的。

（3）压力测试：压力测试就是让系统崩溃的测试，是在超常规负荷条件下，长时间连续运行系统，检验应用程序的各种性能表现和反应。要想方设法给系统添加压力，让系统出现故障，然后观察系统在出现故障时的反应以及故障恢复的能力。例如，系统是缓慢死亡的还是猝死的，是不是保存了崩溃前的数据，故障恢复的时间有多久，恢复之后能不能返回原先的工作状态等等，只要是有利于提高系统在大压力情况下的表现的，都是考察的内容。由于网络环境的复杂性和多样性，压力测试是软件质量保证的重要元素之一，不能马虎了事。

通过测试的网站就可提交客户，交付使用了。

1.3　本章小结

本章作为全书的开始，逐步介绍了静态网站、动态网站的概念，根据开发网站的实践介绍了开发动态网站需要掌握的基本知识领域，尤其着重介绍了网站开发的步骤和基本过程。明确了用户和客户的不同，区别了用户需求和客户需求的差异，强调了系统设计要注意的几个方面。这些多为经验性的总结，实践中没有固定的模式，读者应灵活借鉴，适当参考。从第 2 章开始，就要学习具体的网站开发的相关技术了。

1.4　练习题

1．什么是静态网站和动态网站？两者的主要区别是什么？
2．开发网站需要掌握哪些方面的知识？
3．开发网站的基本步骤是什么？
4．你怎么认识让客户参与开发小组的活动？
5．什么是功能测试？包含几个方面？
6．什么是性能测试？你熟悉性能测试工具吗？

第2章 XHTML 基础

本章重点

- XHTML 与 HTML 的区别；
- XHTML 文件的结构；
- XHTML 基本标记的用法。

2.1 从 HTML 到 XHTML 的演变

超文本标记语言（Hypertext Marked Language，HTML）是一种用来制作超文本文件的简单标记语言。使用 HTML 语言描述的文件能独立于各种操作系统平台而运行在 Web 浏览器上。所谓超文本，是因为它能展示包含文本、图片、声音、动画和影视等内容的多种媒体形式，同时可以通过单击从一个主题跳转到另一个主题，从一个页面跳转到另一个页面与分布在世界各地的主机上的文件链接，直接获取相关的主题，突破了传统访问文件的线性访问方式。

HTML 只是一个纯文本文件。通常，把 HTML 文件或具有这种内容格式的文件称为网页。创建一个 HTML 文件，需要两个工具，一个是 HTML 编辑器；另一个是 Web 浏览器。HTML 编辑器是用于生成和保存 THML 文件的应用程序。Web 浏览器是用来打开 Web 网页文件，提供查看 Web 资源的客户端程序。严格地讲，HTML 算不上一门语言，它只是用来控制网页输出的排版规则。

2.1.1 HTML 的发展历史

1980 年，蒂姆·贝纳斯·李在欧洲粒子物理研究所担任咨询软件工程师的工作。工作过程中，他要频繁地与世界各地的科学家们沟通联系，和他们交换、分析数不清的报告和数据，经常不得不重复回答一些问题，烦琐的过程实在令人烦恼。于是他希望有一种工具，能够让大家不管身处何地，都能通过计算机网络去简单快捷地访问其他人的数据。为此，贝纳斯·李开始在业余时间编写一个软件程序，利用一系列的链接（即超文本）首先将自己计算机上的重要文件的存储地址都“串”起来，这样就可以像从一本书的目录检索到其

中内容那样，通过很简单的操作找到想要的重要文件。他将这个程序命名为“探询（Enquire）”。这个没有发布过的程序就是后来万维网的雏形。它能够存储信息，将文件链接到一起，只能在一台计算机上进行这些操作。贝纳斯・李并不感到满意，希望在此基础上设计一个能够连接全球计算机的超级工具。1990 年，贝纳斯・李开始着手研究万维网（World Wide Web，WWW）项目，创建了一种快速小型超文本语言来为项目服务。WWW 是一个分布式、用于浏览和搜索的系统，能够使用鼠标点击的方式对远程计算机中的文件进行存取，这个系统给互联网中的每一个文件一个唯一的地址，然后使用 HTML 语言对文件的显示进行编码，用浏览器来远程浏览这些文件。1991 年 8 月 6 日，贝纳斯・李建立的第一个网站（也是世界上第一个网站）http://info.cern.ch/上网，它解释了万维网是什么，如何使用网页浏览器和如何建立一个网页服务器等。从此，万维网就开始在互联网上大范围流行起来。

1994 年，贝纳斯・李来到了美国，加入麻省理工学院计算机科学实验室，并就任非营利机构“全球互联网联盟”（World Wide Web Consortium，W3C）组织的执行主席。其成员包括微软、网景、Sun、苹果与 IBM 在内的 160 多家企业，任务是管理和主导全球互联网技术标准，避免任何厂商利用关键技术自行提高网络使用价格与技术难度。

贝纳斯・李被业界公认为“互联网之父”。他的发明改变了全球信息化的传统模式，带来了一个信息交流的全新时代。然而比他的发明更伟大的是，贝纳斯・李并没有像其他人那样为 WWW 申请专利或限制它的使用，而是无偿地向全世界开放，为互联网的全球化普及翻开了里程碑式的篇章，让互联网走进了千家万户。贝纳斯・李的这项发明，加速了信息革命的步伐，推动了知识经济的进程。

2.1.2　从 HTML 到 XHTML

HTML 是学习网页设计的首选入门语言，HTML 从出现到现在，标准在不断完善、功能也越来越强大，但是它的规范化要求不是很严格，仍有很多缺陷和不足。HTML 有一个致命的缺点，就是只适合于人与计算机的交流，不适合计算机与计算机的交流。为此，W3C 开发了 XML 标准，XML（The Extensible Markup Language，可扩展标识语言）是 Web 设计的发展趋势，具有四大特点：优良的数据存储格式、可扩展性、高度结构化以及方便的网络传输。

在 XML 1.0 标准推出时，仍然有大量的人员采用 HTML，而且在万维网中已存在数以百万计的页面是采用 HTML 编写的，还不能直接抛弃 HTML，因此，HTML 的后继者 XHTML 就出现了。XHTML 是可扩展超文本标识语言（The Extensible HyperText Markup Language）。

XHTML 是为了适应 XML 而重新改造的 HTML。它是一种独立的语言，以 HTML 4.01 作为基础，又以 XML 的应用为目的，是从 HTML 到 XML 的过渡。所以，现在学习网页设计，应从 XHTML 开始，如果已经学习过 HTML，那么只要再知道它们的区别就可以了。

2.1.3　XHTML 与 HTML 的区别

XHTML 与 HTML 的区别如下：

（1）XHTML 的标记格式良好，不能重叠，可以嵌套；HTML 语法格式要求不严格。

（2）XHTML 区分大小写，而 HTML 则不区分。

（3）XHTML 的标记必须封闭，如果是空标记，不能如 HTML 一样保持原状，必须加上空格与 / 作为结束。

（4）XHTML 标记的属性必须用=赋予属性值，HTML 一般用属性简写代替。

（5）XHTML 中用 id 属性取代 HTML 中的 name 属性。

2.2 XHTML 文件的结构

一个 XHTML 文件是由一系列的元素和标签组成，元素名不区分大小写。XHTML 用标签来规定元素的属性和它在文件中的位置， XHTML 文件分文件头和文件体两部分，在文件头中，对这个文件进行了一些必要的定义，文件体中才是要显示的各种文件信息。

下面是一个最基本的 XHTML 文件的代码。

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
02 <html xmlns="http://www.w3.org/1999/xhtml">
03 <head>
04 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
05 <title>网页的标题</title>
06 </head>
07 <body>
08     网页内容
09     网页内容
10 </body>
11 </html>
```

说明：

第 1 行是 DOCTYPE 声明，用来声明 XHTML 的版本。DOCTYPE 必须大写，DTD 叫做文件类型定义，包含了文件的规则，浏览器根据定义的 DTD 解释页面的标记并显示。DOCTYPE 声明必须放在每一个 XHTML 文件的最顶部，在所有代码和标记之前。

第 2 行是声明网页使用的名字空间。因为 XHTML 是 HTML 向 XML 过渡的标记语言，它需要符合 XML 的规则，因此也要使用名字空间。由于 XHTML 1.0 不能自定义标记，所以它的名字空间都相同，即 http://www.w3.org/1999/xhtml。

第 4 行定义语言的编码，为了被浏览器正确解释和通过 W3C 代码校验，所以 XHTML 的文件都必须声明它们所使用的编码语言，这里使用的是“gb2312”，即简体中文编码。

XHTML 文件标记的格式为：

```
<html xmlns="http://www.w3.org/1999/xhtml"> 文件内容</html>
```

<html>表示 XHTML 文件的开始，到</html>结束。文件中的所有文本和 XHTML 标签都包含在其中，它表示该文件是以超文本标识语言（XHTML）编写的。事实上，现在常用的 Web 浏览器都可以自动识别 XHTML 文件，并不要求有<html>标签，也不对该标签进行任何操作，但是为了使 XHTML 文件能够适应不断变化的 Web 浏览器，还是应该养成不

省略这对标签的良好习惯。

<head></head>是 XHTML 文件的头部标签，在浏览器窗口中，头部信息是不被显示在正文中的，在此标签中可以插入其他标记，用以说明文件的标题和整个文件的一些公共属性。

<title>和</title>是嵌套在<head>头标记之间的，文本是文件标题，它被显示在浏览器窗口的标题栏上。

<body></body>标记一般不省略，标记之间的文本是正文，是在浏览器中要显示的页面内容。

上面的这几对标签在文件中都是唯一的，head 标签和 body 标签是嵌套在 html 标签中的。

2.3　XHTML 的基本标记

1. <html></html>标记

用于标记文件的开始和结尾。两个标记必须成对使用，网页中其他所有内容都有放在这对标记之间。其格式为：

```
<html>文件内容</html>
```

2. <head></head>标记

它是文件的头标记，用于定义 XHTML 文件的头部信息。两个标记必须成对使用，这对标记之间可使用<title>…</title>、<script>…</script>等标记，以描述 XHTML 文件的相关信息。除了<title>…</title>标记之间的内容，<head>…</head>标记对之间的内容都不会在浏览器的文件窗口显示出来。其格式为：

```
<head>文件头部内容</head>
```

3. <body></body>标记

它是文件的主体标记，用于定义 XHTML 文件的主体部分，两个标记必须成对使用。在这对标记之间定义的是网页上显示的主要内容和其他用于控制文本显示方式的标记。<body></body>标记有许多内置的属性，可以定义页面的背景色彩、背景图像、文字色彩，以及超文本链接颜色等，主要属性如表 2-1 所示。

表 2-1　<body>标记的属性

属　性	描　　述
link	设定页面默认的连接颜色，默认为蓝色
alink	设定鼠标正在单击时的连接颜色，默认为蓝色
vlink	设定访问后连接文字的颜色，默认为蓝色
background	设定页面背景图像
bgcolor	设定页面背景颜色
leftmargin	设定页面的左边距
topmargin	设定页面的上边距
bgproperties	设定页面背景图像为固定，不随页面的滚动而滚动
text	设定页面文字的颜色，默认为黑色

4. <title></title>标记

它是文件的标题标记，用于浏览器标题栏上的文本信息显示。其格式为：

```
<title>标题</title>
```

5. <!……-->注释标记

<!--表示注释的开始，-->表示注释的结束。

6. <hr/>标记

<hr/>标记用于在页面中画一条水平线段，线段的样式由其属性决定，主要属性如表 2-2 所示。

表 2-2　<hr/>标记的属性

属性	参　数	功　能	单　位	默 认 值
size		设置水平分隔线的粗细	pixel（像素）	2
width		设置水平分隔线的宽度	pixel（像素）、%	100%
align	left center right	设置水平分隔线的对齐方式		center
color		设置水平分隔线的颜色		black
noshade		取消水平分隔线的 3d 阴影		

7. <div></div>标记

它是一个区块容器标记，在<div></div>之间可以容纳段落、标题、表格、图片等各种 XHTML 元素。

程序 2-1.html 演示了这几个基本标记的应用。

程序 2-1.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 2-1.html，我的第一个 html 程序-->
<head>
  <title>2-2.html</title>
</head>
<body>
 <div align="center" width="250" height="50">
 这是我的第一个 html 程序，我很高兴哦。
 <hr size="2" color="blue" width="250" />
 </div >
 <div>
  这个程序简单了点．学习嘛，要一步一步地来，不着急。
 </div>
</body>
</html>
```

程序运行结果如图 2-1 所示。



图 2-1　程序 2-1.html 运行结果

2.4　文本排版

1. <hx></hx>（x=1,…,6）标记

标题元素由标签<h1>到<h6>定义。<h1>定义了最大字号的标题，<h6>定义了最小字号的标题。

2. <p></p>段落标记

段落标记定义了同一个段落的文字。不同段落间的间距等于连续加了两个换行符，也就是要隔一行空白行，用以区别文字的不同段落。下一个<P>的开始就意味着上一个<P>的结束。

换行标记
，也叫空标签，不包含任和内容，在 XHTML 文件中的任何位置只要使用了
标签，当文件显示在浏览器中时，该标签之后的内容将显示下一行。

3. <pre></pre>预排版标记

要保留原始文字排版的格式，就可以通过<pre>标签实现，方法是把制作的文字排版内容前后分别加上始标签<pre>和尾标签</pre>。在此标记中不能使用<img>标记来插入图像，也不能使用<object>标记来插入 ActiveX 控件或 Java applet。

4. 文字格式标记

用于控制文字的字体、大小和颜色。控制方式是利用属性进行设置。font 标签的属性如表 2-3 所示。

表 2-3　<font>标记的属性

属　性	使 用 功 能	默 认 值
face	设置文字使用的字体	宋体
size	设置文字的大小	3
color	设置文字的颜色	黑色

程序 2-2.html 演示了这几个标记的排版作用。

程序 2-2.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<!--程序 2-2.html，我的第二个 html 程序-->
<head>
  <title>2-1.html</title>
</head>
<body>
 <div  width="250" height="50">
 <h2>系统设计的总体规划目标</h2>
<p> 宣传国家的相关法规政策，发布物流资讯，
<font face="黑体" color="red" size="4">探索设计实现物流领域的第四
方物流、<br/>虚拟物流、物流金融等行 业先进理念，满<br/>足客户对物流服务的多样性
需求。
</font></p>
 </div >
 <div>
 <pre>
 ①线性可伸缩性。可持续增长以满足用户需求和业务复杂性。
  ②持续的服务可用性。使用冗余和功能专业化来提高容错能力。
    ③数据和基础结构的安全性。保护数据和基础结构免受恶意攻击或盗用。
    ④管理的简便性和完整性。确保运作能够满足增长的需求。
  ⑤灵活的可重用性。系统组件可多次重复使用。
 ⑥系统的开放性。开放性设计节省投资成本。
</pre>
 </div>
</body>
</html>
```

程序运行结果如图 2-2 所示。



图 2-2 程序 2-2.html 运行结果

2.5 超链接

Web 上的网页是互相链接的，单击被称为超链接的文本或图形就可以链接其他页面。建立超链接的标签为<a>和。格式为：

```
<a href="资源地址" target="窗口名称" title="指向连接显示的文字">超链接名称</a>
```

属性 href 定义了这个链接所指的目标地址；

属性 target 用于指定打开链接的目标窗口，其默认方式是原窗口。其取值及含义如表 2-4 所示。

表 2-4 属性 target 的值及含义

属 性 值	描 述
_parent	在上一级窗口中打开，一般使用分帧的框架页会经常使用
_blank	在新窗口打开
_self	在同一个帧或窗口中打开，这项一般不用设置
_top	在浏览器的整个窗口中打开，忽略任何框架

属性 title 用于指定指向链接时所显示的标题文字。

创建指向其他服务器的页面链接格式为：

```
<a href="http://server/path/目标文件名.html>热点文本</a>
```

创建一个指向页面特定部分的链接格式为：

```
<a href="#记号名">热点文本</a> (链接到本页内)
<a href="http://server/path/目标文件名.html>热点文本</a>（链接到其他服务器上页面的某个指定位置）
```

创建一个指向电子邮件的链接格式为：

```
<a href=mailto:username@domain>热点文本</a>
```

创建一个指向下载文件的链接格式为：

```
<a href="可下载文件名”>热点文本</a>
```

程序 2-3(1).html 是简单超链接例子。

程序 2-3(1).html

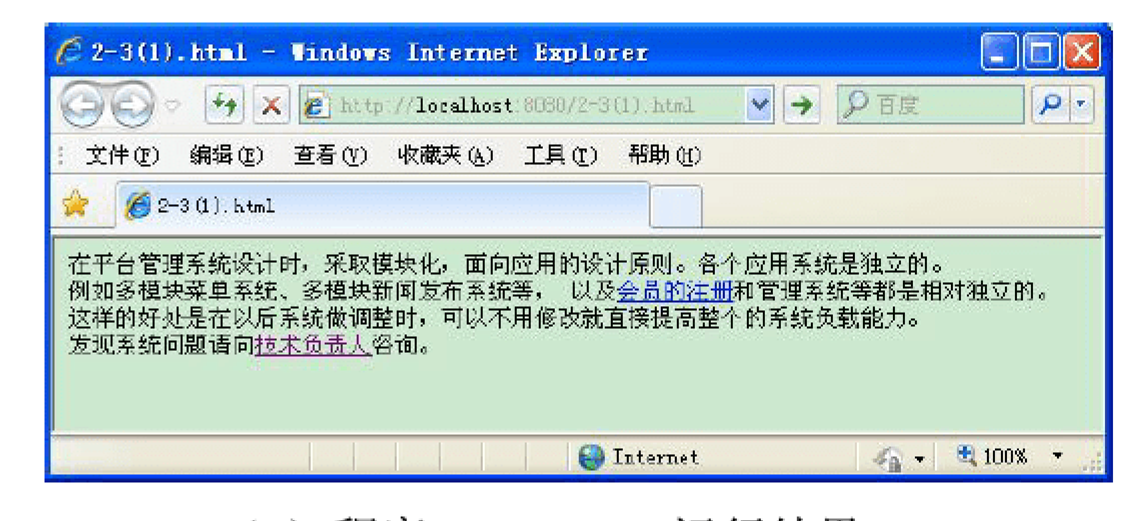
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--程序 2-3(1).html，超链接-->
<head>
  <title>2-3(1).html</title>
</head>
<body>
在平台管理系统设计时，采取模块化，面向应用的设计原则。各个应用系统是独立的。<br/>
例如多模块菜单系统、多模块<a href="2-3(2).html" target="_blank" title="会员注册">新闻发布</a>系统等，
以及会员的注册和管理系统等都是相对独立的。<br/>这样的好处是在以后系统做调整时，可以不用修改就直接提
高整个的系统负载能力。<br/>发现系统问题请向<a href=mailto:jingxiuzhao@126.com>技术负责人</a>咨询。
</body>
</html>
```

程序 2-3(2).html 可见给出单击程序 2-3(1).html 中的超链接“新闻”后的页面。

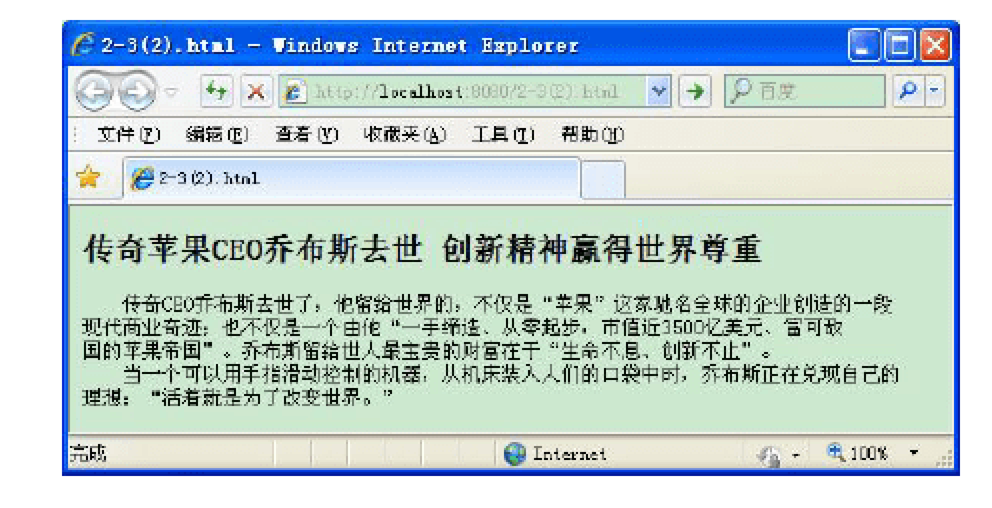
程序 2-3(2).html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--程序 2-3(2).html，超链接->新闻浏览-->
<head>
  <title>2-3(2).html</title>
</head>
<body>
<h2>传奇苹果 CEO 乔布斯去世 创新精神赢得世界尊重</h2>
<p>
传奇 CEO 乔布斯去世了，他留给世界的，不仅是“苹果”这家驰名全球的企业创造的一段<br/>
现代商业奇迹；也不仅是一个由他“一手缔造、从零起步，市值近 3500 亿美元、富可敌<br/>
国的苹果帝国”。乔布斯留给世人最宝贵的财富在于“生命不息、创新不止”。<br/>
当一个可以用手指滑动控制的机器，从机床装入人们的口袋中时，乔布斯正在兑现自己的<br/>
理想：“活着就是为了改变世界。”
</p>
</body>
</html>
```

程序 2-3(1).html 和程序 2-3(2).html 运行结果分别如图 2-3（a）和图 2-3（b）所示。



（a）程序 2-3(1).html 运行结果



（b）程序 2-3(2).html 运行结果

图 2-3 运行结果

2.6 图像和多媒体

网页中插入图片用单标记<img/>，当浏览器读取到<img/>标记时，就会显示此标记所设定的图像。<img/>标记还可以播放 avi 格式的视频。通过使用标记的属性，可对插入的图片、视频进行修饰、控制，其可用属性如表 2-5 所示。

表 2-5 <img/>标记的属性

属 性	描 述
src	图像的 url 的路径
alt	提示文字
width	宽度，通常只设为图片的真实大小以免失真，改变图片大小最好用图像工具

续表

属　　性	描　　述
height	高度，通常只设为图片的真实大小以免失真，改变图片大小最好用图像工具
dynsrc	avi 文件的 url 的路径
loop	设定 avi 文件循环播放的次数
loopdelay	设定 avi 文件循环播放延迟
start	设定 avi 文件的播放方式
lowsrc	设定低分辨率图片，若所加入的是一张很大的图片，可先显示图片
usemap	映像地图
align	图像和文字之间的排列属性
border	边框
hspace	水平间距
vlign	垂直间距

网页中播放 Flash 动画、MP3 音乐、电影等多媒体内容用<embed></embed>标记，播放控制使用如表 2-6 所示的属性。

表 2-6　<embed>标记的属性

属　　性	描　　述
src	多媒体文件的 url 的路径
width	播放窗口的宽度
height	播放窗口的高度
loop	控制是否循环播放，取值 true 时无限次播放，false 只播放一次
autostart	控制多媒体是否循环播放，取值 true 自动播放，false 不自动播放
hidden	控制播放面板的显示和隐藏，取值 true 时隐藏，false 时显示

程序 2-4.html 演示了<img/>标记的用法，多媒体标记的用法请自行尝试。

程序 2-4.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--程序 2-4.html，图像和多媒体-->
<head>
  <title>2-4.html</title>
</head>
<body>
<h3>不朽的程序员</h3>
<div style="float:left"><image src="Knuth.gif"height="60" width="100"/>
 <image src="art.jpg"height="60" width="100"></div>
<font size="2">Donald Ervin Knuth,1938 年 1 月 10 日出生,英文名直译为唐纳德·欧
文·克
```

```
努特，"高德纳"这个中文名字是 1977 年他访问中国之前，计算机科学家姚期智的夫人姚储枫
给他起的。术语"算法"(Algorithm)和"数据结构"(Data Structure)就是克努特于 29 岁时
提出来的。他的名著《计算机程序设计艺术》被列为 20 世纪最佳 12 部学术专著之一，与狄拉
克的"量子力学"、爱因斯坦的"相对论"、曼德布罗特的"分形论"、鲍·林的"化学键"、罗素和怀特海德的"数学基础"、冯·诺依曼和摩根斯坦的"博弈论"、维纳的"控制论"、伍德沃和霍夫曼的"轨道对称性"、费曼的"量子电动力学"等科学史上的重要著作并列必读经典。</font>
</body>
</html>
```

程序运行结果如图 2-4 所示。

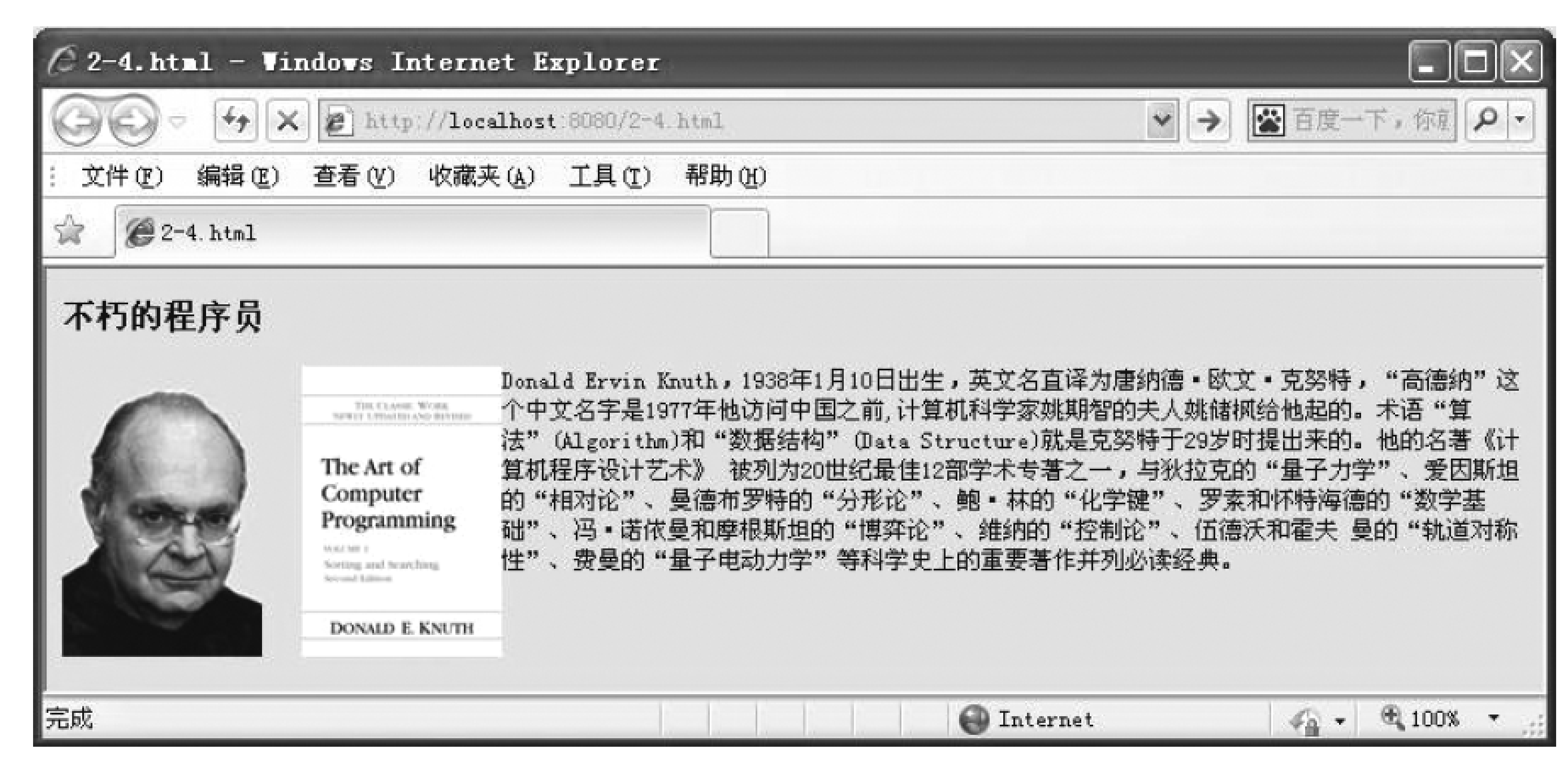


图 2-4 程序 2-4.html 运行结果

2.7 列表

列表是分段排出一组级别相同的项目。XHTML 提供了三种不同的列表：无序列表、有序列表和定义列表。

1. 无序列表标记

无序列表使用的一对标签是
，无序列表指没有进行编号的列表，每一个列表项前使用<li>。<li>的属性 type 有三个选项，这三个选项都必须小写：

disc 表示实心圆。

circle 表示空心圆。

square 表示小方块。

如果不使用其项目<li>的属性值，即默认情况下的<ul>会加上"实心圆"。

2. 有序列表标记

有序列表和无序列表的使用格式基本相同，使用标签
，每一个列表项前使用<li>。<ol>列表的结果是带有前后顺序之分的编号。如果插入和删除一个列表项，编号会自动调整。

顺序编号的设置是由<ol>的两个属性 type 和 start 来完成的。start=编号开始的数字，如 start=2 则编号从 2 开始，如果从 1 开始可以省略，或是在<li>标签中设定 value＝"n"改变列表行项目的特定编号，如<li value="7">。type=用于设定编号的数字、字母等的类型，如 type=a，则编号用英文字母。为了使用这些属性，把它们放在<ol>或<li>的初始标签中。

有序列表 type 的属性如表 2-7 所示。

表 2-7　属性“tgpe”的值及含义

type	描　　述
type=1	表示列表项目用数字标号（1,2,3…）
type=A	表示列表项目用大写字母标号（A,B,C…）
type=a	表示列表项目用小写字母标号（a,b,c…）
type=I	表示列表项目用大写罗马数字标号（Ⅰ,Ⅱ,Ⅲ…）
type=i	表示列表项目用小写罗马数字标号（i,ii,iii…）

3. <dl></dl>定义列表标记

自定义列表以 <dl> 标记开始。每个自定义列表项以 <dt> 开始。每个自定义列表项的定义以 <dd> 开始。定义性列表可以用来给每一个列表项再加上一段说明性文字，说明独立于列表项另起一行并缩排显示。程序 2-5.html 演示了三种不同列表的应用。

程序 2-5.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">0304 <html xmlns=
   "http://www.w3.org/1999/xhtml">
<!--程序 2-5.html:列表-->
<head>
      <title>2-5.html</title>
</head>
     <body>
     <div type="circle" >
<h4>这是无序列表</h4>
<ul >
      <li>数据结构</li>
      <li>操作系统</li>16  <li>人工智能</li>
      <li>网络工程</li>18  </ul>
     </div>
<div >
     <h4>这是有序列表</h4>
<ol type="a" >
 <li>数据结构</li>
 <li>操作系统</li>
 <li>人工智能</li>
 <li>网络工程</li>
```

```
        </ol>
       </div>
       <div >
        <h4>这是定义列表</h4>
     <dl>
       <dt>计算机专业课程</dt>
       <dd>数据结构</dd>
       <dd>操作系统</dd>
       <dt>软件工程专业课程</dt>
       <dd>软件测试</dd>
       <dd>软件建模</dd>
     </dl>
       </div>
   </body>
   </html>
```

程序运行结果如图 2-5 所示。

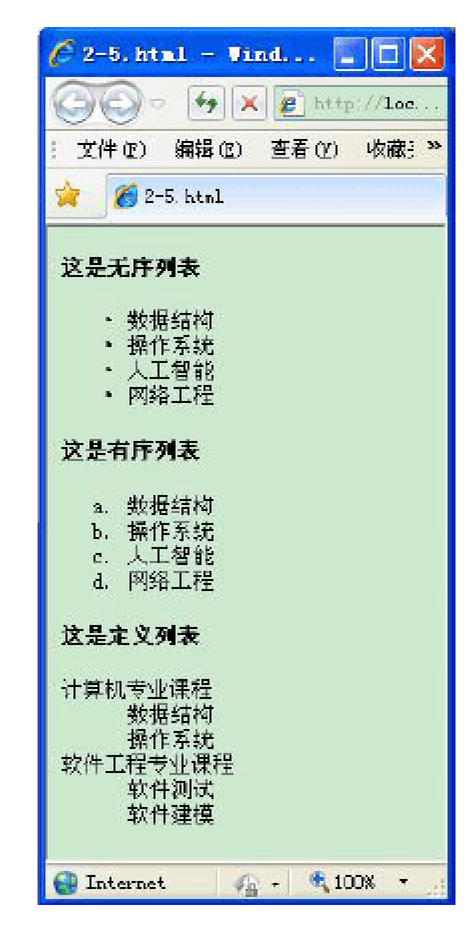


图 2-5　程序 2-5.html 运行结果

2.8 表格

表格的应用十分广泛，表格是通过<table>,<th>,<tr>,<td>标记来完成的，各标记含义如表 2-8 所示。

表 2-8　表 格 标 记

标　　记	描　　述
<table>…</table>	用于定义一个表格开始和结束
<th>…</th>	定义表头单元格。表格中的文字将以粗体显示，在表格中也可以不用此标签，<th>标签必须放在<tr>标签内
<tr>…</tr>	定义一行标签，一组行标签内可以建立多组由<td>或<th>标签所定义的单元格
<td>…</td>	定义单元格标签，一组<td>标签将将建立一个单元格，<td>标签必须放在<tr>标签内

在一个最基本的表格中，必须包含一组<table>标记，一组标记<tr>和一组<td>标记或<th>。控制表格显示的常用<table>标记的属性如表 2-9 所示。

表 2-9　<table>标记的属性

属　　性	描　　述
width	表格的宽度
height	表格的高度
align	表格在页面的水平摆放位置
background	表格的背景图片
bgcolor	表格的背景颜色
border	表格边框的宽度（以像素为单位）
bordercolor	表格边框颜色

续表

属　　性	描　　述
bordercolorlight	表格边框明亮部分的颜色
bordercolordark	表格边框昏暗部分的颜色
cellspacing	单元格之间的间距
cellpadding	单元格内容与单元格边界之间的空白距离的大小

表格中的每一行都是由<tr></tr>标记定义的，行的属性如表 2-10 所示。

表 2-10　<tr>标记的属性

属　　性	描　　述
bgcolor	行的背景颜色
background	行的背景图像文件的 URL 地址
align	行中单元格的水平对齐方式
valign	行中单元格内容的垂直对齐方式取值为 top、middle、bottom、baseline
bordercolor	行的边框颜色
bordercolorlight	行的 3D 边框的高亮显示颜色
bordercolordark	行的 3D 边框的阴影颜色

表格中每行有若干个单元格，单元格通过<td></td>标记定义，标题单元格可用<th></th>标记定义。它们的属性如表 2-11 所示。

表 2-11　单元格的属性

属　　性	描　　述
align	单元格的水平对齐方式
bgcolor	单元格的背景颜色
bordercolor	单元格的边框颜色，在<table>标记的 border!=0 有效
bordercolordark	单元格的 3D 边框的阴影颜色
bordercolorlight	单元格的 3D 边框的高亮显示颜色
colspan	合并单元格时一个单元格跨越的表格列数
rowspan	合并单元格时一个单元格跨越的表格行数
valign	单元格中文本的垂直对齐方式
nowrap	单元格中的文本不换行

程序 2-6.html 演示了简单表格的应用。

程序 2-6.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--演示控制表格的标记的应用-->
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 2-6.html:表格-->
<head>
```

```
<title>2-6.html</title>
</head>
<body>
<table align=center border=1 bordercolor=#CC6235>
<caption><font size=5 color=blue>进货单</font></caption>
    <tr bgcolor=#CCCCCC>
      <th width=80>部门</th>
      <th width=80>货号</th>
      <th width=80>货名</th>
     <th width=90>数量</th>
     <th width=90>产地</th>
    </tr><tr>
     <td rowspan=3><font color=blue>超市</font></td>
     <td>sm-1101</td>
     <td>白 糖</td>
     <td align=center>180</td>
     <td align=center>南宁</td>
    </tr><tr>
     <td>sm-1107</td>
     <td>牙 膏</td>
     <td align=center>90</td>
     <td align=center>天津</td>
    </tr> <tr>
     <td>sm-1122</td>
     <td>保温杯</td>
     <td align=center>83</td>
     <td align=center>临沂</td>
    </tr><tr>
     <td><font color=green>家电部</font></td>
     <td>fe-5201</td>
     <td>液晶电视</td>
     <td align=center>82</td>
     <td align=center>青岛</td>
    </tr>
    </table>
    </body>
    </html>
```

程序运行结果如图 2-6 所示。



部门	货号	货名	数量	产地
超市	sm-1101	白 糖	180	南宁
	sm-1107	牙 膏	90	天津
	sm-1122	保温杯	83	临沂
家电部	fe-5201	液晶电视	82	青岛

图 2-6 程序 2-6.html 运行结果

2.9　表单

表单提供了重要的人机交互功能，用<form></form>标记创建。

表单是由窗体和控件组成的，表单是个容器，它能够容纳各种各样的控件。常用的控件有文本框、密码框、下拉菜单、单选框和复选框等。控制表单的属性如表 2-12 所示。

表 2-12　<form>标记的属性

属　　性	描　　述
name	表单的名称，可用于引用或控制该表单
id	指定表示该标记的唯一标识码
action	接收表单提供的数据的处理程序的程序名（包括绝对或相对路径）
method	属性用来定义处理程序从表单中获得信息的方式，取值为 get,post
target	属性用来指定目标窗口，取值为_top,_blank,_parent,_self

action 属性值是空值（""）时，数据传给当前文件。

method 属性值为 get 时，数据以 url 方式传递，即表单数据附加在 url 后面传递，这种方式传送的数据量一般限制在 1KB 以下。method 属性取值为 post 时，传送的数据量要比使用 GET 方式的大得多。

表单标记的格式：

```
<form action="url" method=get|post name="myform" target="_blank">…</form>
```

表单中可以放置许多控件，常用的有<input>、<select>、<textarea>等。以下仅对最常用的控件给予介绍。

1. <input/>输入控件标记

在 XHTML 语言中，标记<input>具有重要的地位，能够将浏览器中的控件加载到 XHTML 文件中。该标记是单个标记，没有结束标记。<input type="">标记用来定义一个用户输入区，用户可在其中输入信息。此标记必须放在 <form></form>标记对之间。<input type="">标记中提供了 9 种类型的输入区域，具体的类型由 type 属性的值决定。<input>标记的属性如表 2-13 所示。

表 2-13　<input>标记的属性

属　　性	描　　述
type	定义输入控件类型，取值为 text, checkbox, password, radio, button, hidden, submit, reset, image 等
name	定义表单控件的名称，submit 和 reset 除外，服务器通过调用某一输入区域的名字的 value 值来获得该区域的数据
size	定义表单域的长度
marxlength	表示该文本输入框允许用户输入的最大字符数
checked	为 radio,checkbox,button 类型设置是否选中
value	为 button,reset,submit 定义按钮上的文本信息

续表

属　　性	描　　述
disabled	除 hidden 外的其他类型，第一次装载时，用户不能执行写或选择操作
onchang	当文本改变时要执行的函数
onselect	当控件被选中时要执行的函数
onfocus	当文本接受焦点时要执行的函数
src	插入图像的地址
align	指定对齐方式，可取 top, bottom, middle

2. <select></select>下拉列表框标记

<select></select>标记对用来创建一个菜单下拉列表框。此标记对用于<form></form>标记对之间。<select>具有 multiple、name 和 size 属性。multiple 属性不用赋值，直接加入标记中即可使用，加入了此属性后列表框就成了可多选的了；name 是此列表框的名字，它与上边讲的 name 属性作用是一样的；size 属性用来设置列表的高度，缺省时值为 1，若没有设置（加入）multiple 属性，显示的将是一个弹出式的列表框。

<option>标记用来指定列表框中的一个选项，放在<select></select>标记对之间。此标记具有 selected 和 value 属性，selected 用来指定默认的选项，value 属性用来给<option>指定的那一个选项赋值，这个值是要传送到服务器上的，服务器正是通过调用<select>区域的名字的 value 属性来获得该区域选中的数据项的。各项属性如表 2-14 所示。

表 2-14　<select>标记的属性

属　　性	描　　述
name	列表框名字，作识别之用
size	列表框高度，默认为 1
multiple	允许做多项选择
selected	指定默认选项

3. <textarea></textarea>多行文本框标记

<textarea></textarea>用来创建一个可以输入多行的文本框，此标记对用于<form></form>标记对之间。<textarea>属性如表 2-15 所示。

表 2-15　<textarea>标记的属性

属　　性	描　　述
name	文字区块的名称，用作识别之用
rows	文字区块的列数，即其高度
cols	文字区块的宽度
onblur	当控件失去焦点时要执行的函数
onchang	当文本改变时要执行的函数
onselect	当控件被选中时要执行的函数
onfocus	当文本接受焦点时要执行的函数
wrap	定义输入内容大于文本域时显示的方式，取值为 off，virtual，Physical

wrap 的默认值是文本自动换行。当输入内容超过文本域的右边界时会自动转到下一行，数据在被提交处理时自动换行的地方不会有换行符出现。

取值为 off 时，当输入的内容超过文本域右边界时，文本将向左滚动，文本不会换行。

取值为 virtual，允许文本自动换行，而数据在被提交处理时自动换行的地方不会有换行符出现。

取值为 physical，文本允许换行，当数据被提交处理时换行符也将被一起提交处理。

程序 2-7.html 演示了表单的简单应用。

程序 2-7.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 2-7.html:表单-->
<head>
     <title>2-7.html</title>
</head>
<body>
<form name="form1" method="post" action="">
  <table width="400" border="0" align="center" bgcolor="#FFFFFC">
 <tr>
 <td colspan="2" bgcolor="CFCCCC"><div align="center">公司员工个人信息</div></td>
    </tr>
    <tr>
      <td>工 号:</td>
      <td><input name="GH" type="text" value="11-78101"></td>
    </tr>
    <tr>
      <td>姓 名：</td>
      <td><input name="XM" type="text" value="万顺风"></td>
    </tr>
    <tr>
      <td>性 别：</td><td>
        <input name="SEX" type="radio" value="男" checked="checked">男
        <input name="SEX" type="radio" value="女">女</td>
    </tr>
    <tr>
      <td>出生日期：</td>
      <td><input name="BD" type="text" value="1978-05-06"></td>
    </tr>
    <tr>
      <td>所学专业：</td>
      <td><select name="ZY">
        <option>机械制造</option>
        <option>软件工程</option>
        <option>信息管理</option>
      </select></td>
    </tr>
    <tr>
```

```
      <td>所属部门：</td>
      <td><select name="BM" size="3" multiple>
        <option selected>安全部</option>
        <option>一车间</option>
        <option>质检部</option>
        <option>二车间</option>
        <option>信息部</option>
      </select></td>
    </tr>
<tr>
 <td>家庭情况：</td>
 <td><textarea name="JTQK" rows="5" >山东青岛即墨人，农村户口，一子一女，年总收入 7 万元</textarea></td>
</tr>
<tr>
 <td>特 长：</td>
 <td><input name="TC" type="checkbox" value="篮球" checked="checked" >篮球
 <input name="TC" type="checkbox" value="电焊">电焊
<input name="TC" type="checkbox" value="产品推广" checked="checked">产品推广</td>
      </tr>
      <tr>
        <td><input type="submit" name="BUTTON1" value="提交"></td>
        <td><input type="reset" name="BUTTON2" value="重置"></td>
      </tr>
 </table>
 </form>
</body>
</html>
```

程序运行结果如图 2-7 所示。

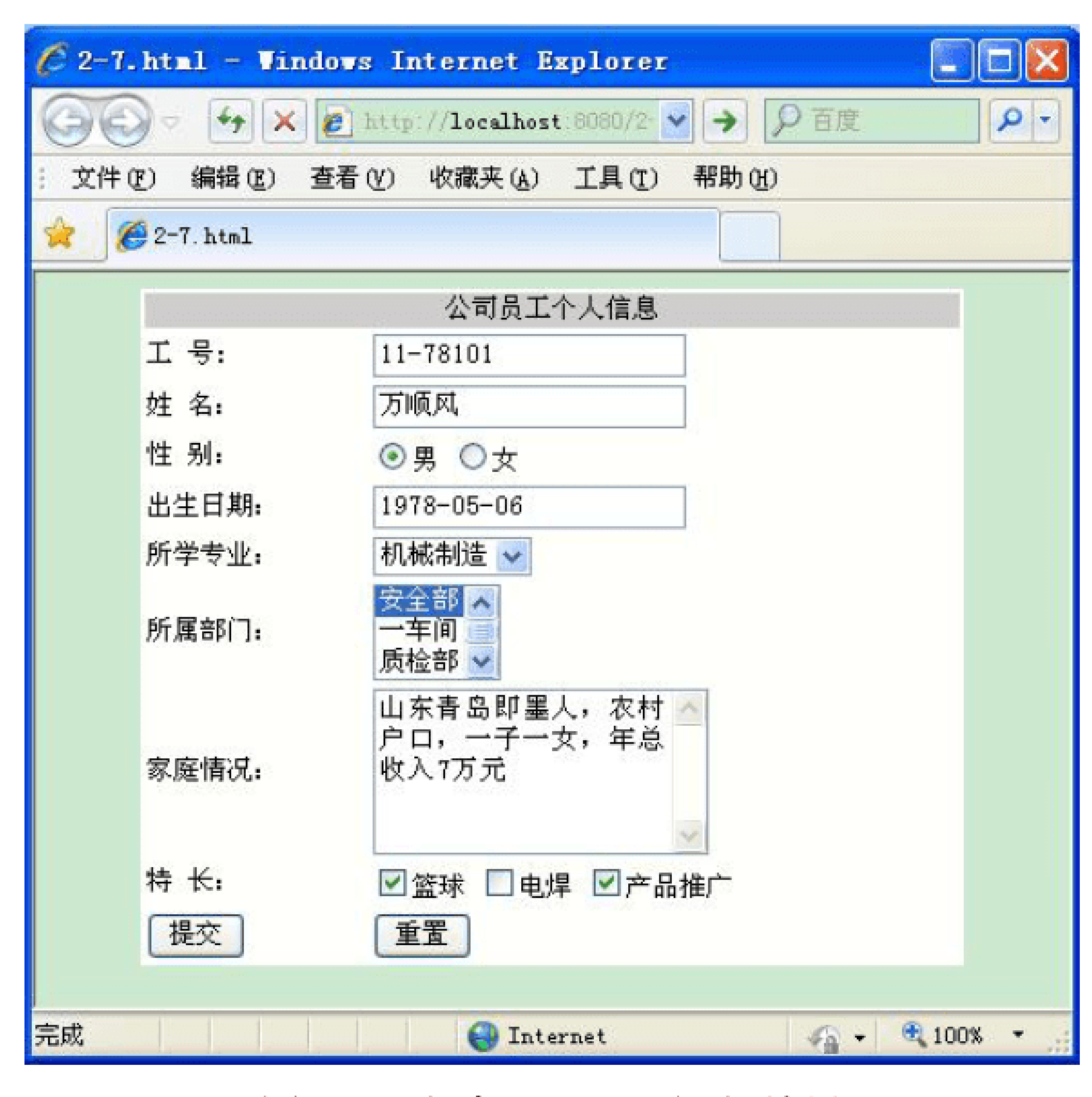


图 2-7 程序 2-7.html 运行结果

2.10 框架

框架就是把一个浏览器窗口划分为若干个小窗口，每个窗口可以显示不同的 URL 网页。使用框架可以非常方便地在浏览器中同时浏览不同的页面。标记<frameset></frameset>和<frame></frame>组合可以定义框架。其中，frameset 定义框架的容器，frame 定义框架中的子窗口。控制 frameset 的属性如表 2-16 所示。

表 2-16 <frameset>标记的属性

属　性	描　述
border	设置边框粗细，默认是 5 像素
bordercolor	设置边框颜色
frameborder	指定是否显示边框，0 代表不显示边框，1 代表显示边框
cols	用“像素数”和%分割左右窗口，*表示剩余部分
rows	用“像素数”和%分割上下窗口，*表示剩余部分
framespacing="2"	表示框架与框架间的保留空白的距离
noresize	设定框架不能调节，只要设定了前面的，后面的将继承

<frame>是个单标记，<frame>标记要放在框架集 frameset 中，<frameset>设置了几个子窗口就必须对应几个<frame>标记，而且每一个<frame>标记内还必须设定一个网页文件，控制 frame 的属性如表 2-17 所示。

表 2-17 <frame>标记的属性

属　性	描　述
src	指示加载的 url 文件的地址
bordercolor	设置边框颜色
frameborder	指示是否要边框，1 显示边框，0 不显示（不提倡用 yes 或 no）
border	设置边框粗细
name	指示框架名称，是连接标记的 target 所要的参数
noresize	指示不能调整窗口的大小，省略此项时就可调整
scorlling	指示是否要滚动条，auto 根据需要自动出现，yes 有，no 无
marginwidth	设置内容与窗口左右边缘的距离，默认为 1
marginheight	设置内容与窗口上下边缘的边距，默认为 1
width	框窗的宽及高，默认为 width="100" height="100"
align	可选值为 left, right, top, middle, bottom

子窗口的排列遵循从左到右，从上到下的次序规则。下面的程序演示了框架的应用，共 4 个文件组成。

主框架页面（程序 2-8frame.html）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 2-8frame.html:框架-->
<head>
<title>2-8fram.html</title></head>
<frameset rows="100, *">
    <frame src="2-8top.html" name="frmtop">
    <frameset cols="150,*">
        <frame src="2-8left.html" name="frmleft" >
        <frame src="2-8content.html" name="frmmain" >
    </frameset>
</frameset>
</html>
```

框架顶部页面：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">0304
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 2-8top.html:-->
<body bgcolor="#FF88FF">
  <img src="2-8t.jpg" width="600" height="60"/>
</body>
</html>
```

框架左侧页面：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 2-8left.html-->
<head>
<title>2-8left.html</title></head>
<body>
 <a href="http://www.dtqzjt.com/index2.php" target="frmmain">公司主页</a><br><br>
 <a href="2-7.html" target="frmmain">员工信息</a><br><br>
 <a href="http://www.baidu.com" target="frmmain">百度搜索</a><br>
</body>
</html>
```

框架右侧页面：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 2-8content.html:-->
<head>
```

```
<title>2-8-content.html</title></head>
<body>
    <h2 >欢迎来德通公司喝茶！</h2>
    <img src="2-8c.jpg"/>
</body>
</html>
```

程序运行结果如图 2-8 所示。



图 2-8　程序 2-8.html 运行结果

2.11　XHTML 的语法规则

XHTML 的作用与 HTML 相同，但在语法规则上要比 HTML 严格，其语法规则大致如下：

（1）与 HTML 不同，XHTML 文件必须通过 DOCTYPE 声明文件类型定义。

（2）<html>标记声明中必须指定“xmlns="http://www.w3.org/1999/xhtml" ”属性的 XML 名称空间。

XML 名称空间标识 XHTML 文件使用的标记范围，XHTML DTD 定义的标记不与用户定义的标记或其他 DTD 中定义的标记冲突。值得注意的是，许多 XHTML 文件中并没有声明 xmlns="http://www.w3.org/1999/xhtml"属性，但是运行时浏览器会将这个属性自动加入到代码中。

（3）XHTML 基于 XML，XHTML 必须具有良好的格式。XHTML 标记与标记之间可以嵌套，但是不能重叠。即嵌套形如
…
是合法的，但是重叠形式如
是非法的。

（4）XHTML 中的所有元素必须是封闭的，如果元素本身是非封闭的，将给标记的尾

端添加一个空格和“/”，以保持 XHTML 与浏览器的兼容。

如，HTML 4.01 的换行标记
，在 XHTML 就要写成
。

（5）XHTML 中的所有标签和属性必须是小写形式。

例如：

```
<hr/>
```

（6）name 属性在 XHTML 已经没有定义，取而代之的是用 id 属性来确定标记（如 a、img、applet、map、frame、form 等）的名称。

例如：

```
<a id="ab">kkkkk</a>
```

XHTML 的页面结构：

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>…</title></head>
<body>…</body>
</html>
```

两种页面结构对比如下：

```
<!-- HTML 文件 -->
<html>
<head>
<title>html 实例</title>
</head>
<body>
<form action="" method="get">
   <p>single select
   <input TYPE=radio checked>
   </p>
</form>
</body>
</html>
```

```
            <!--XHTML 文件 -->
      <!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
      strict.dtd">
      <html>
      <head>
      <title>xhtml 实例</title>
      </head>
      <body>
      <form action="" method="get">
         <p>single select
         <input type="radio" checked=
         "checked" />
         </p>
      </form>
      </body>
      </html>
```

2.12 本章小结

在 HTML 4.0 的基础上，用 XML 的规则对其进行扩展，得到了 XHTML，XHTML 是由 W3C 发布的，是 Web 标准家族的一部分。XHTML 负责网页的结构、表现和行为三个部分中的结构部分，是一个向 XML 过渡的技术。XML 是 Web 发展的趋势，但面对成千上万已有的基于 HTML 语言设计的网站，直接采用 XML 还为时过早，目前应该采用的就是

XHTML。与 HTML 相比，XHTML 的语法严密，主要表现在 XHTML 元素一定要被正确地嵌套使用，XHTML 文件一定要有正确的组织格式，标记名要用小写字母，所有的 XHTML 元素一定要关闭，就是独立的标签也要用“/>”结束，属性名字必须小写，属性值必须写在引号内，属性的缩写被禁止，用 id 属性代替 name 属性等。

如果在学习过程中自己编写了一个符合标准的站，你可以到 http://validator.w3.org/check?uri=http%3A%2F%2Fmoban.7880.com%2F 进行验证。若不成功则会出现红色警示，并会告知哪里有错误，可以依照指出的错误慢慢修改。

2.13　练习题

1．为什么要用 XHTML？它与 HTML 的主要区别有哪些？

2．一个最基本的 XHTML 文件需要哪些标记？

3．如何建立有序表、无序表？

4．如何建立一个表格？如何在表格中嵌套表格？

5．如何建立表单？常用的表单控件有哪些？

6．常用的多媒体标记有哪些？

7．什么是框架？框架有什么作用？它和表格有什么区别？

8．创建一个 XHTML 文件，其内容为自己的姓名、班级、手机号、学习的课程、自己的照片和 E-mail 地址。文件中要包含<strong>、</hr>、<p>和
标签。

9．创建一个 XHTML 文件，带有一个表单，表单中包含以下控件：一个文本框、4 个复选框、一组单选按钮、一个下拉列表。

10．创建一个 XHTML 文件，设计成个人家乡的主页。文件中至少包含 3 个图片和相应的文字描述、一个有序表、一个表格。

第3章 CSS+DIV

本章重点

- CSS 的概念及优点；
- CSS 嵌入 XHTML 的方法；
- CSS 的属性及应用；
- DIV 的概念及优点；
- DIV 的定位方法。

3.1 什么是 CSS 和 DIV

进行 Web 开发要遵循一定的标准，Web 标准是一系列标准的集合。网页主要由 3 部分组成：结构（Structure）、表现（Presentation）和行为（Behavior）。对应的标准也分为 3 个方面：结构化标准，主要包括 XHTML 和 XML；表现标准主要包括 CSS；行为标准主要包括对象模型（如 W3C DOM）和 ECMAScript 等。这些标准大部分由 W3C 起草和发布，也有一些是其他标准组织制订的标准，如 ECMA（European Computer Manufacturers Association）的 ECMAScript 标准。

第 2 章已经介绍了控制内容的结构标准 XHTML，现在该介绍控制内容表现方式的层叠样式表 CSS 了。层叠样式表（Cascading Style Sheets，CSS），是用于控制网页样式并允许将样式信息同网页内容分离的一种标记性语言。在网页设计中，XHTML 负责组织网页的结构和内容，CSS 负责决定页面的表现形式。W3C 于 1996 年 12 月发布了 CSS 1.0 规范，1998 年 5 月发布了 CSS 2.0 规范。目前最新版本是 CSS 3.0 版。W3C 并不强制要求软件厂商的产品必须符合其规范，所以目前流行的浏览器都没有完全符合 CSS 的规范，这给网页设计师带来了浏览器兼容的问题。但是 CSS 可实现对网页的外观和排版的精确控制，可对多个元素统一设置，可取代 XHTML 表格样式的布局。CSS 布局与 XHTML 相结合能帮助设计师分离外观与结构，便于页面的修改，便于页面风格的统一，减少网页的体积，使站点的访问及维护更加容易，极大地提高了工作效率。

在 Web 标准中，除了用 CSS 控制内容的表现方式外，还用 DIV 作为容纳内容的基本

单元。DIV 是不含任何语义的标记，用来在其中放置任何网页元素，是个块级容器。

页面中的所有元素都可以看成一个盒子，占据着一定的页面空间。可以通过调节盒子的边框和距离等参数，来调节盒子的位置。DIV 当然也是盒子，一个盒子模型由内容（Content）、边框（Border）、内边距（Padding）、外边距（Margin）这 4 个部分组成，如图 3-1 所示。

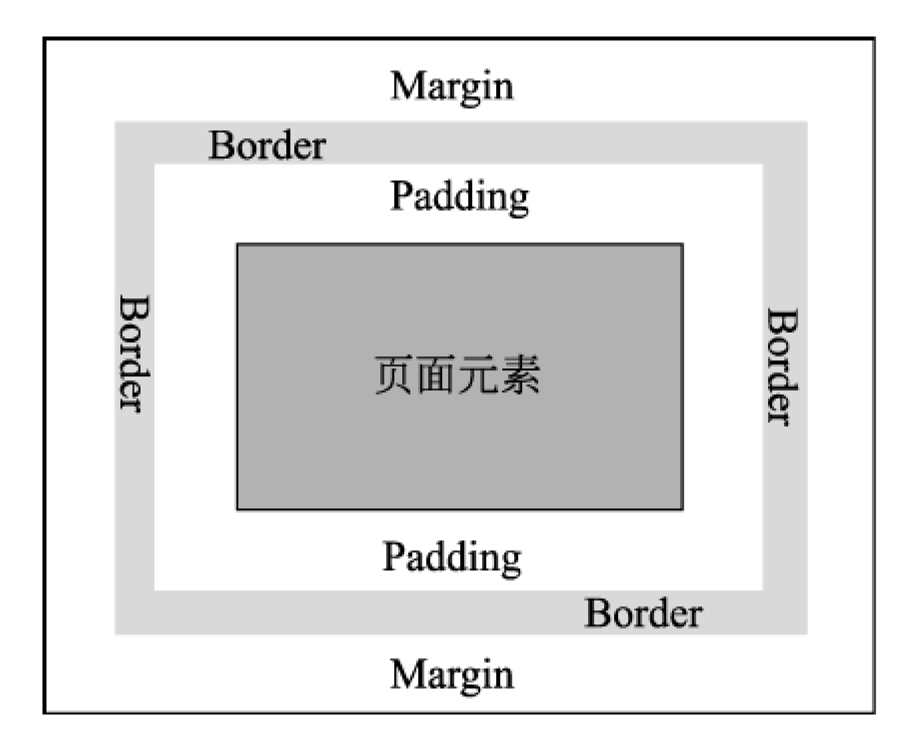


图 3-1　DIV 组成示意图

CSS+DIV 布局就是用 DIV 将页面分块，用 CSS 设计各块的位置大小以及相互关系，在页面的各大 DIV 中插入作为各个栏目框的小 DIV。CSS+DIV 布局符合 W3C 标准，目前已成为网页布局的主流。

3.1.1　CSS 基础

CSS 的思想就是首先指定对什么“对象”进行设置，然后指定对该对象的哪个方面的“属性”进行设置，最后给出该设置的“值”。因此，概括来说，CSS 就是由“对象”、“属性”和“值”3 个基本部分组成的。

CSS 定义的基本语法格式为：

```
选择符{规则列表}
```

选择符是指要使用该样式的对象，指定了对哪些网页元素进行设置。CSS 选择符主要有 XHTML 标记选择符、CLASS 选择符、ID 选择符和伪类选择符等 4 种。它可以是一个或多个 XHTML 标记、CLASS 选择符或 ID 选择符，如果为多个则使用逗号“，”进行分隔。

规则列表是由一个或多个属性定义语句组成的样式规则，各语句间使用分号“;”进行分隔。

属性定义语句的语法格式为：

“属性名：属性值”

例如：

```
h3{font-family:隶书;color:blue}
h2,h3{ font-family:宋体;color:red }
myfont{font-size:20pt}
#myfont{font-size:20pt}
```

3.1.2　在 XHTML 中使用 CSS 的方法

在 XHTML 中引入 CSS 文件，使得 CSS 描述作用于 XHTML 中的内容，产生设计者的设计效果。在 XHTML 中引入 CSS 的方法有以下 4 种。

1. 行内式

这种方式可以直接在 XHTML 标记中定义该标记的显示样式，并且该样式定义只能用于这个标记。其语法格式为：

```
<标记名 style="样式属性名 1：属性值 1；
              样式属性名 2：属性值 2；…">
```

例如：

```
<p style= " font-family:宋体;color:red; font-size:10pt " >
```

在 XHTML 1.1 中，W3C 不建议使用行内式，因为将来可能不予支持。

2. 嵌入式

这种方式通过<style>标记来定义样式，其语法格式为：

```
<style type= "text/css" >
<!--
   选择符 1，选择符 2，…{样式属性名 1：属性值 1；
                      样式属性名 2：属性值 2；…}
⋮
-->
</style>
```

其中使用了 XHTML 注释标记<!--…-->，是为了当有浏览器不支持 CSS 语句时，遇到该语句段就会忽略该段内容。

3. 导入式

为了让多个页面文件可以共享 CSS 样式定义，可以将 CSS 语句段编写为单独的一个 CSS 文件，然后将它嵌入到页面文件中。其语法格式为：

```
<style type= "text/css" >
<!--
@import url("外部 CSS 样式表文件名");
-->
</style>
```

其中，“外部 CSS 样式表文件名”是以.css 为扩展名的文件，如果该文件和当前页面文件处于同一目录，则直接给出文件名；否则给出其相对路径。

4. 链接式

这种方式和嵌入外部样式表的方式相似，也要访问外部样式表，但是嵌入外部样式表时是将样式文件直接加载到 import 语句处，而链接外部样式表是直接向样式文件索取样式。其语法格式为：

```
<link type= "text/css" rel=stylesheet  href="外部 CSS 样式表文件名">
```

以上 4 种引入 CSS 的方法，对应于 3 种不同层次的样式表：行内样式表、文档样式表和外部样式表。之所以称为层叠样式表，是因为它们可以在 3 个层次上进行定义。当多个样式冲突时，从低到高为行内样式表优先于文档样式表，文档样式表又优先于外部样式表。即较低层的样式表能覆盖较高层的样式表，一个标签内容的样式是通过样式表应用的层次来确定的。

行内样式表只能作用于单个标签的内容，这种样式表指定的样式信息是以与 XHTML 完全不同的语言进行描述，但又是和 XHTML 混在一起的。CSS 旨在将网页的样式与内容分开描述，使得文档变得更加清晰，因此在 XHTML 1.1 中 W3C 不建议使用行内式样式表。

文档样式表说明位于文档的头部区域，能够作用于整个文档主体。这是一种为文档所

有内容的呈现提供统一样式的方式。以嵌入式引入的样式表就是文档样式表。

外部样式表能够作用于多个文档。外部样式表独立于文档存在，彻底地将显示内容和显示的样式分开为不同的文件。当需要使用外部样式表时，只要在需要的文档的头部以导入式或链接式的方式引入所需的样式表即可。

3.1.3　选择符的分类

CSS 选择符主要有 XHTML 标记选择符、CLASS 选择符和 ID 选择符和伪类选择符 4 种。选择符的定义和使用方法如表 3-1 所示。

表 3-1　选择符的定义和使用方法

选择符	语法格式	样式使用范围说明	示　　例
XHTML 标记	定义语法：标记{…} 使用语法：<标记>	在 XHTML 文件中，所有该标记包含的文本都具有定义的 CSS 样式	h3{font-family:隶书} …
CLASS 选择符	定义语法：*.类名{…} 或　.类名{…} 使用语法：<标记 class=类名>	在 XHTML 文件中的所有使用该类名的标记都具有定义的 CSS 样式	.my{font-size:20pt} <h3 class=my>…</h3>
	定义语法：标记.类名{…} 使用语法：<标记 class=类名>	在 XHTML 文件中的所有指定该类名的该标记都具有定义的 CSS 样式	h3.my{font-size:20pt} <h3 class=my>…</h3>
ID 选择符	定义语法：#ID 名{…} 使用语法：<标记 id=ID 名>	在 XHTML 文件中的所有使用该 ID 名的标记都具有定义的 CSS 样式	#my{font-size:20pt} <h3 id=my>…</h3>
	定义语法：标记#ID 名{…} 使用语法：<标记 id=ID 名>	在 XHTML 文件中的所有指定该 ID 名的该标记都具有定义的 CSS 样式	h3#my{font-size:20pt} <h3 id=my>…</h3>
伪类选择符	定义语法： 选择符:伪类 {…}	在 XHTML 文件中的所有指定伪类的元素都具有定义的 CSS 样式	a:link { color: red } a:visited{ color: green} a:hover{color:white} a:active { color: blue}

当有多个选择符需要设置相同的属性和属性值时，可以用逗号将选择符隔开进行组合定义，即一次性设置多个选择符的属性和属性值，这样可以减少样式重复定义。

```
selector1, selector2,…{property1: value;…}
```

包含选择符是另外一种形式的组合定义，例如：

```
li a
{
   font-size: 12px
}
```

li 和 a 之间没有通过逗号隔开，该样式定义的是，在列表内的链接的文字大小为 12 像素，而列表外的链接的文字仍为默认大小。

3.1.4 第一个 CSS 文件

下面以外部样式表初步演示 CSS 的用法。首先制作 CSS 文件 style.css。打开文本编辑器，如记事本，输入下面的代码，然后以 style1.css 名保存。

```
body
  {
    background-color: yellow;
    background-image: url("3-1b.jpg");
    background-position:200px 100px;
    background-repeat:repeat-x;
  }
h1
  {
    color:#FC9C66;
    background-color: #FCCC04;
    text-align:center;
  }
p
  {
    font-size:20px;
    font-style:italic;
    color:blue;
 }
```

再制作网页文件程序 3-1.html，文件中引入上面保存的文件 style1.css。

程序 3-1.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 3-1.html:第一个样式表-->
  <head>
     <title>3-1.html</title>
     <link rel="stylesheet" type="text/css" href="style1.css" />
  </head>
  <body>
  <h1>这是第一个使用 CSS 样式表的程序，使用的是外部样式表</h1>
  <p>CSS 的思想就是首先指定对什么“对象”进行设置，然后指定对该对
   象的哪个方面的“属性”进行设置，最后给出该设置的“值”。 因此，概
   括来说，CSS 就是由 3 个基本部分——“对象”、“属性”和“值”组成的。</p>
  </body>
</html>
```

程序运行结果如图 3-2 所示。

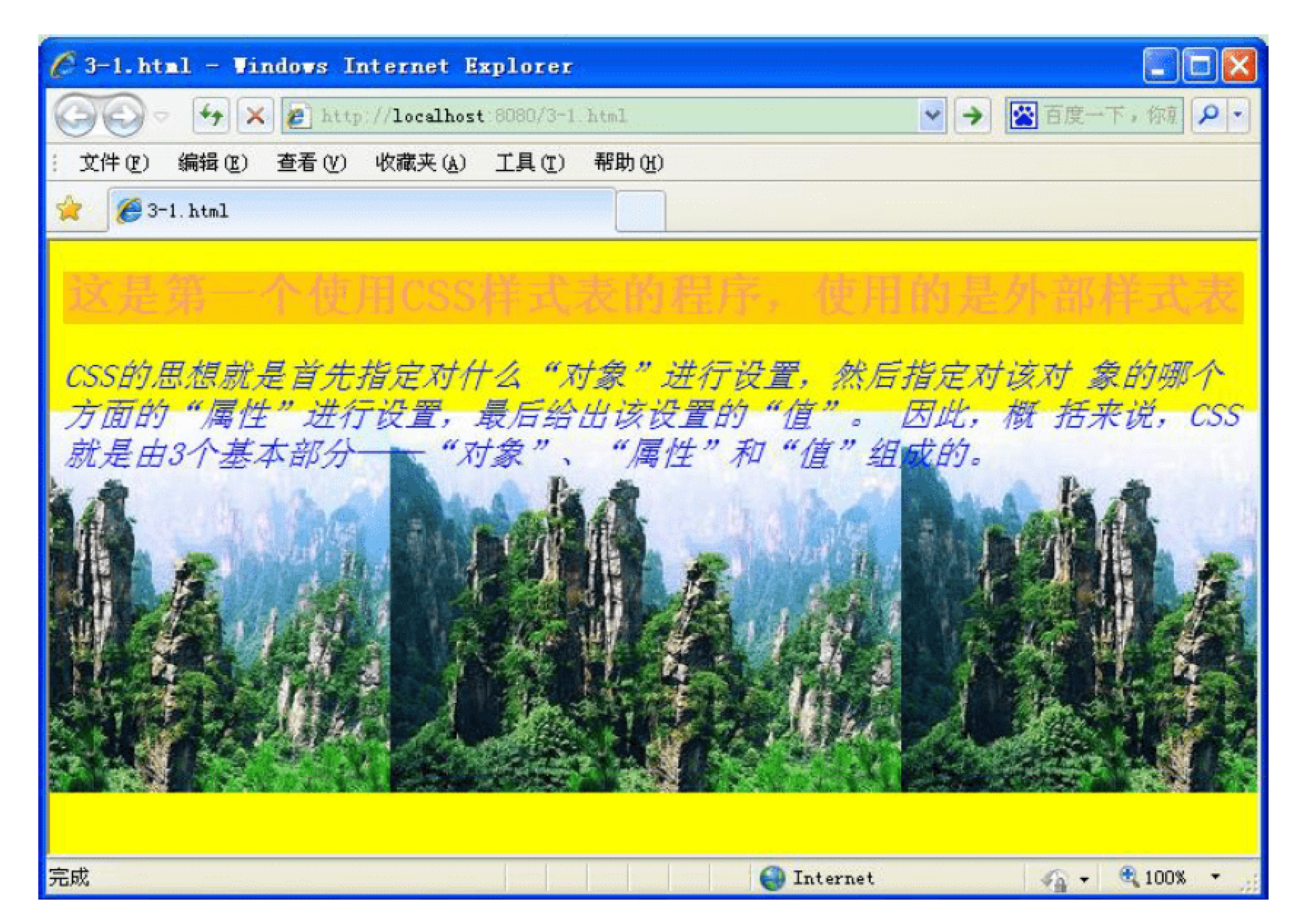


图 3-2　程序 3-1.html 运行结果

3.2　CSS 的属性

CSS 中包含了 60 多种属性，可分为背景、文本、字体、列表、边界、边框、边距以及定位 8 类。

3.2.1　背景属性

恰当地使用背景，能更好地衬托要表达的主题，提供网页更丰富的视觉效果。控制背景的属性如表 3-2 所示。

表 3-2　背景属性

属　性	属 性 值	说　明
background	其他背景属性值的集合	复合属性，用于综合设置背景。该属性不可继承
background-color	transparent，color	设置背景颜色
background-image	none，url (url)	设置背景图像
background-attachment	scroll，fixed	设置背景图像是随对象内容滚动还是固定的
background-repeat	repeat，no-repeat，repeat-x，repeat-y	设置背景图像的平铺方式
background-position	top，center，bottom，left，center，right，x% y%，x-pos，y-pos	设置图像显示的起始位置

背景属性可按以下格式简写：

```
background : background-color || background-image || background-repeat ||
background-attachment || background-position
```

默认值为 transparent none repeat scroll 0% 0%。程序 3-2.html 以嵌入的方式引入 CSS 文档文件，演示了简单背景控制的情况。

程序 3-2.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 3-2.html:背景属性-->
  <head>
    <title>3-2.html</title>
<style type="text/css">
            h1
            {
            text-align:center;
                color:red;
                font-size:30px;
            }
            body
            {
                  background-image:url(3-2b.jpg);
                  background-position:200px 20px;
                  background-repeat:no-repeat;
            }
            p
            {
                font-size:16pt;
                color:#C0F;
            }
        </style>
  </head>
  <body>
  <h1>内容简介</h1>
    <p>
《人月神话(英文版)》内容源于作者 Brooks 在 IBM 公司任 System/360 计算机系列以及
其庞大的软件系统 OS/360 项目经理时的实践经验。在《人月神话(英文版)》中，Brooks 为
人们管理复杂项目提供了最具洞察力的见解，既有很多发人深省的观点，又有大量软件工
程的实践，为每个复杂项目的管理者给出了自己的真知灼见。 </p>
  </body>
</html>
```

程序运行结果如图 3-3 所示。



图 3-3　程序 3-2.html 运行结果

3.2.2　文本属性

文本属性主要用于块标签中文本的样式设置，常用的属性有缩进、对齐方式、行高、文字与字母间隔、文本转换与文本修饰等。控制文本的属性如表 3-3 所示。

表 3-3　文本属性

属　　性	属　性　值	说　　明
text-align	left，right，center，justify	设置对象中文本的水平对齐方式
text-decoration	none 默认值，underline，overline，line-through	设置对象中文本的装饰效果
text-indent	指定长度或百分比	设置对象中文本块首行缩进
line-height	数字或百分比	设置行高
word-spacing	数字默认值为 0，可为负	设置文字间隔
letter-spacing	数字默认值为 0，可为负	设置字母或字符间隔
vertical-align	baseline，sub，super，top，text-top，middle，bottom，text-bottom，length	设置文本垂直对齐
text-transform	uppercase，lowercase，capitalize	设置字母大小写的转换控制

其中部分属性的含义如下：

baseline：将支持 valign 特性的对象的内容与基线对齐。

sub：垂直对齐文本的下标。

super：垂直对齐文本的上标。

top：将支持 valign 特性的对象的内容与对象顶端对齐。

text-top：将支持 valign 特性的对象的文本与对象顶端对齐。

middle：将支持 valign 特性的对象的内容与对象中部对齐。

bottom：将支持 valign 特性的对象的文本与对象底端对齐。

text-bottom：将支持 valign 特性的对象的文本与对象顶端对齐。

length：CSS2 由浮点数字和单位标识符组成的长度值或者百分数。可为负数。定义由基线算起的偏移量。基线对于数值来说为 0，对于百分数来说就是 0%。

word-spacing 默认值为 0。word-spacing 的值可以为负数。当 word-spacing 的值为正数时，文字之间的间隔会增大；反之，word-spacing 的值为负数时，文字间距就会减少。

程序 3-3.html 演示了文本属性的控制。

程序 3-3.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 3-3.html:文本属性-->
<head>
  <title>3-3.html</title>
     <style type="text/css">
     /*文本属性设置*/
         h3{text-family:黑体}
     .d1{line-height:20px;word-spacing:8px; text-indent:30px
                ;text-align:left;text-transform:lowercase;margin:auto}
     .d2{color:red}
     .d3{color:blue;text-decoration:underline}
 </style>
</head>
<body>
  <div class=d1>
  <h3>C 语言之父丹尼斯·里奇</h3>
   <p>丹尼斯·里奇(Dennis Ritchie)出生于 1941 年 9 月 9 日，是<span class=d3>哈佛大
      学</span>数学博士及著名的美国计算机科学家，为计算机的 C 语言等编程语言以
      及 Multics 和 UNIX 等操作系统作出了重大贡献，被业界誉为<span class=d2>“C 语
      言之父”和“UNIX 之父”</span>。他于 1983 年与肯·汤普逊(Ken Thompson)
      共同获得<span class=d2>图灵奖</span>，理由是<span class=d3>“研究并发
      展通用性操作系统 UNIX”</span>。 </p>
  <p>丹尼斯·里奇先生于 2011 年 10 月 8 日在位于美国<span class=d3>新泽西州</span>
  的家中病逝。计算机爱好者们以特有的方式纪念这位编程语言的重要奠基人。许多网友的发
  帖中没有片言只字，仅仅留下一个分号“;”。</p>
  </div>
</body>
</html>
```

程序运行结果如图 3-4 所示。



图 3-4　程序 3-3.html 运行结果

3.2.3　字体属性

文字属性用来设置网页中文字的显示效果，主要包括文字颜色、字体、文字加粗、字号、文字样式。控制字体的属性如表 3-4 所示。

表 3-4　字体属性

属　性	属　性　值	说　明
color	颜色值	指定颜色
font-family	字体名	设置字体名称
font-size	absolute-size，relative-size，length	设置对象中的字体尺寸
font-style	normal，italic，oblique	设置对象中的字体样式
font-weight	normal，bold，bolder，lighter，number	设置或检索对象中的文本字体的粗细

程序 3-4.html 演示了对字体的控制。

程序 3-4.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 3-4.html:字体属性-->
<head>
<title>3-4.html</title>
<style type="text/css">
/*文字属性设置*/
h3{font-family:隶书;font-weight:bolder;color:green;margin:auto}
.d1{font-family: 幼 圆 ;font-size:12px;font-style:normal;color:blue;
font-weight:bold}
.d2{font-family:楷书;font-size:14px;font-style:italic;color:#0000;
    font-weight:bolder}
    </style>
```

```
    </head>
    <body>
        <div>
        <h3>世人还是更看重眼前</h3>
        <p class=d1>
C 语言的开发是科技史上不可磨灭的伟大贡献。苹果、微软，以及其他公司，都是站
在里奇的肩膀上。缺少了里奇所创造的 C 语言和 UNIX，网络和任何网络产品都
不可能存在。由此可见，我们怎样评价里奇先生的丰功伟绩都不足为过。
    </p>
 <p class=d2> 回忆一周之前，乔布斯去世时，网络上铺天盖地诸多赞誉与哀思，
其产品风靡全球所带来的用户崇拜史无前例。享受到了声势浩大的追思。相比
之下，里奇先生对当代科技进程做出了更大的贡献，可公众甚至不知道他是谁，
这十分不公平。其实，里奇先生更应享受这些赞誉，甚至更多。这足以说明，世
人还是更看重眼前。
        </p>
        </div>
    </body>
</html>
```

程序运行结果如图 3-5 所示。

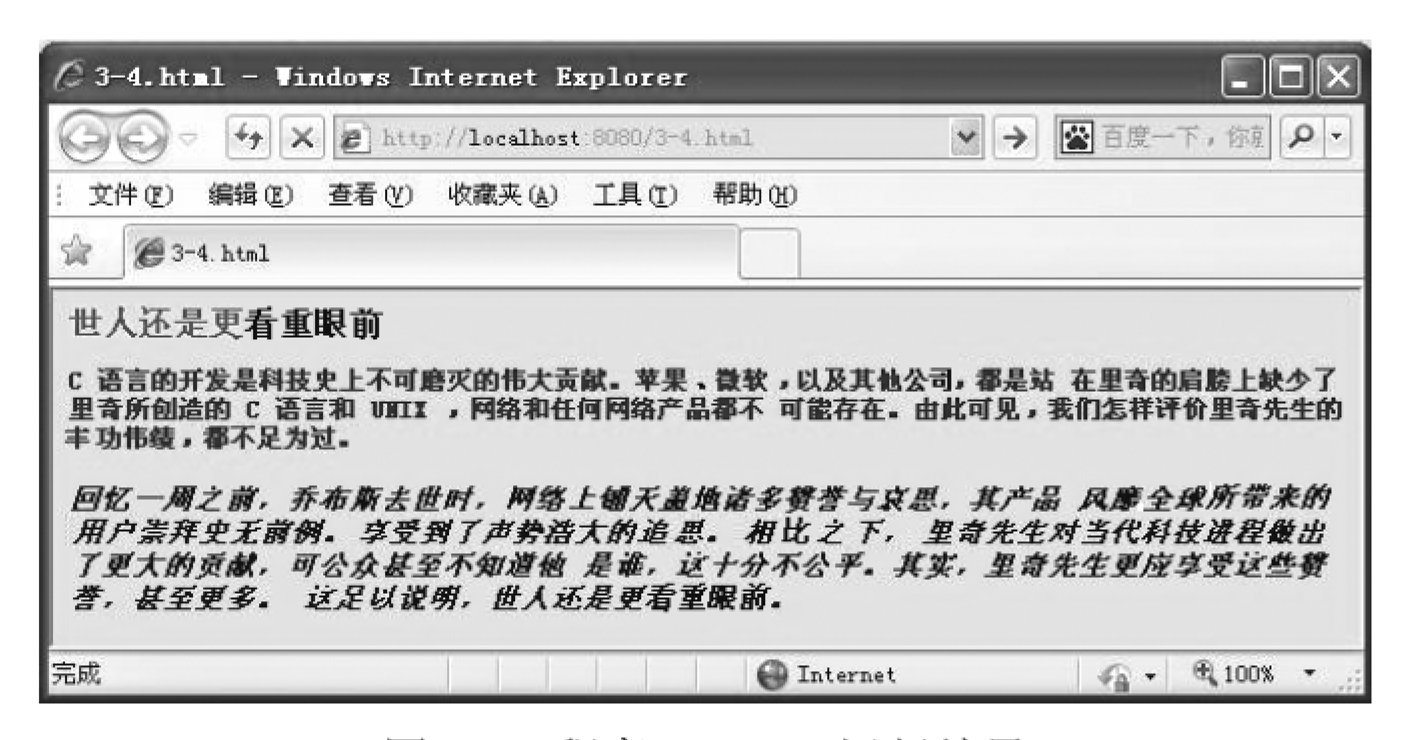


图 3-5 程序 3-4.html 运行结果

3.2.4 边框属性

边框属性用来设置对象边框的颜色、样式和宽度。使用边框属性的对象，先设定对象的 height 或 width 属性，或设定 position 属性为 absolute。

边界的位置见图 3-1。控制边框的属性如表 3-5 所示。

表 3-5 边框属性

属　　性	属　性　值	说　　明
border	按顺序设置宽度、样式和颜色	设置所有边框属性的简写
border-color	color	设置边框颜色
border-style	none，hidden，dotted，dashed，solid，double，groove，ridge，inset，outset	设置边框风格
border-width	medium，thin，thick，length	设置边框宽度

边框的复合属性定义方式为 border : border-width || border-style || border-color。如使用该复合属性定义其单个参数，则其他参数的默认值将无条件覆盖各自对应的单个属性设置。默认值为 medium none。border-color 的默认值将采用文本颜色。

注意：在使用 border-color、border-width 和 border-style 时，如果提供全部 4 个参数值，将按上—右—下—左的顺序作用于 4 个边框。如果只提供一个，将用于全部的 4 条边。如果提供两个，第一个用于上—下，第二个用于左—右。如果提供 3 个，第一个用于上，第二个用于左—右，第三个用于下。

程序 3-5.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 3-5.html:边框属性-->
<head>
<title>3-5.html</title>
<style type="text/css">
h3{font-family:隶书;font-weight:bolder;color:green}
/*边框属性设置*/
.div1{  width:190px;
       height:145px;
       background-color:#FFFFCC;
    border-top-color:blue;
    border-top-style:groove;
    border-top-width:10px;
    border-bottom-color:aqua;
    border-bottom-style:ridge;
    border-bottom-width:10px;
    border-left-color:blueviolet;
    border-left-style:inset;
    border-left-width:10px;
    border-right-color:darkmagenta;
    border-right-style:solid;
    border-right-width:10px;
    }
img{width:188px;height:140px;}
  </style>
</head>
<body>
<h3>丹尼斯·里奇</h3>
   <div class=div1>
     <img src="Dennis Ritchie.jpg"/>
   </div>
</body>
</html>
```

程序运行结果如图 3-6 所示。

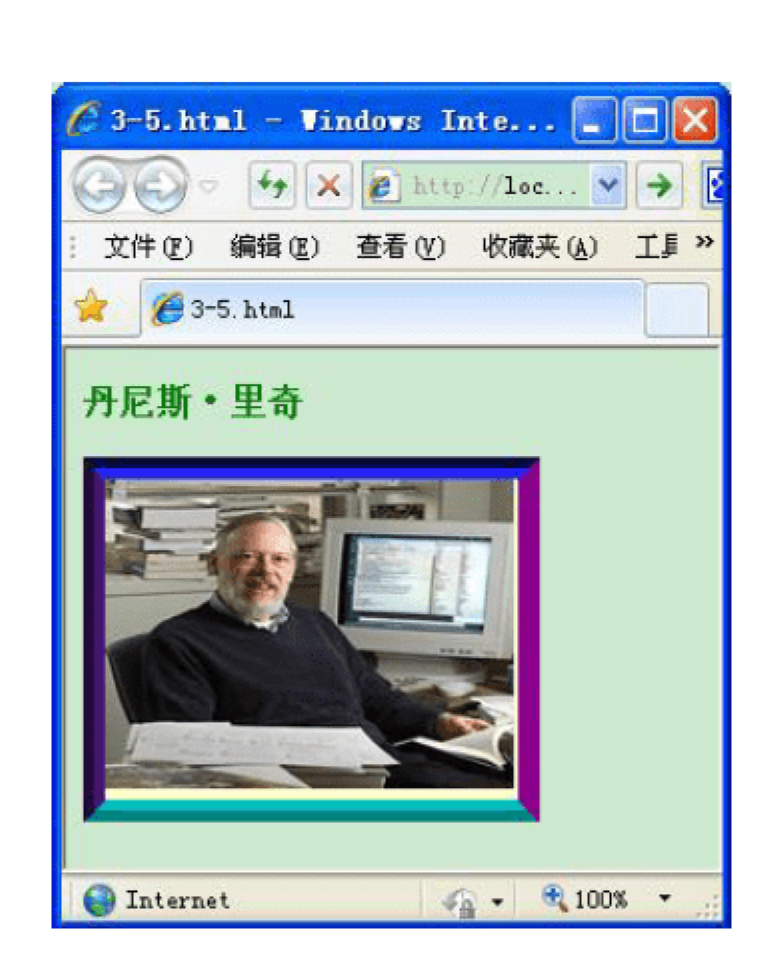


图 3-6　程序 3-5.html 运行结果

3.2.5 外边距属性

使用边界属性的对象，先设定对象的 height 或 width 属性，或设定 position 属性为 absolute。边界的位置见图 3-1。控制外边距的属性如表 3-6 所示。

表 3-6 外边距属性

属　　性	属 性 值	说　　明
Margin	auto，length	设置边界属性
Margin-top	auto，length	设置对象的上外边距
Margin-right	auto，length	设置对象的右上外边距
Margin-bottom	auto，length	设置对象的下外边距
Margin-left	auto，length	设置对象的左外边距

注意：在使用 margin 属性时，如果提供全部 4 个参数值，将按上—右—下—左的顺序作用于 4 边。如果只提供一个，将用于全部的 4 边。如果提供两个，第一个用于上—下，第二个用于左—右。如果提供 3 个，第一个用于上，第二个用于左—右，第三个用于下。margin 值可为负值。

3.2.6 内边距属性

使用边距属性的对象，先设定对象的 height 或 width 属性，或设定 position 属性为 absolute。内边距 Padding 的位置见图 3-1。控制内边距的属性如表 3-7 所示。

表 3-7 内边距属性

属　　性	属 性 值	说　　明
Padding	length	设置对象四边的内边距
Padding-top	length	设置对象四边的上内边距
Padding-right	length	设置对象四边的右内边距
Padding-bottom	length	设置对象四边的下内边距
Padding-left	length	设置对象四边的左内边距

如果提供全部 4 个参数值，将按上—右—下—左的顺序作用于四边。如果只提供一个，将用于全部的 4 条边。如果提供两个，第 1 个用于上—下，第 2 个用于左—右。如果提供 3 个，第 1 个用于上，第 2 个用于左—右，第 3 个用于下。padding 不允许负值。

程序 3-6.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
```

```
<!--程序 3-6.html:边距属性-->
<head>
<title>3-6.html</title>
  <style type="text/css">
  .div1{width:300px;height:240px;border-style:solid;border-color:red;
border-width:1px;}
  .div2{width:360px;height:190px;border-style:solid;border-color:red;
border-width:1px;}
  img{ border-style:solid;border-color:#999;border-width:2px;
  margin-top:15px;margin-bottom:16px;margin-left:5px;margin-right:12px;
  padding-top:5px;padding-bottom:5px;padding-left:12px;padding-
  right:12px;
  width:260px;height:190px;background-color:yellow;}
 .cls2{text-align:justify;width:300px;height:150px;background-color:
AliceBlue;
   border-style:solid;border-color:blue;border-width:1px;
    margin-top:10px;margin-bottom:10px;margin-left:8px;margin-right:
12px;
  padding-top:10px;padding-bottom:5px;padding-left:10px;padding-right:
10px}
  </style>
</head>
 <body>
   <h3>赫伯特·西蒙</h3>
   <hr height="1"width=300px; align="left" color="green"  />
   <div class=div1>
   <img src="jsj.jpg"/>
  </div>
 <hr height="1"width=350px; align="left" color="green"/>
  <div class=div2>
<p class=cls2>管理学大师西蒙(Herbert Alexander Simon1916- 2001)与中国的关
系十分密切,
他先后来中国访问交流达 10 次之多。除了他的祖国以外，西蒙在中国呆过的时间是最长的。他
同中国的多个大学和研究机构有着多方面的学术合作。学界对西蒙有着业余外交家之称，
而这位业余外交家的主要对象，就是中国。1978 年由于西蒙对"经济组织内的决策过程
进行的开创性的研究"，荣获诺贝尔经济学奖。1975 年他和艾伦·纽厄尔因为在人工智能、
人类心里识别和列表处理等方面进行的基础研究，荣获图灵奖。</p>
  </div>
  <hr height="1"width=350px; align="left" color="green" />
</body>
</html>
```

程序运行结果如图3-7所示。读者应对照div的结构图，仔细观察border、margin、padding在图中所对应的位置。



图 3-7　程序 3-6.html 运行结果

3.2.7　定位和 float 属性

定位属性主要从定位方式、层叠顺序、与父标签的相对位置等 3 个方面设置。float 就是让设置的标签产生浮动效果。正常情况下，XHTML 页面中块元素都是从上倒下排列的。如果想实现左右结构，设置 DIV 的 float 属性就是必需的一种选择。控制定位的属性如表 3-8 所示。

表 3-8　定位属性

属性	属　性　值	说　　明
z-index	auto,numbe 可为负数	设置对象的层叠顺序
position	static，absolute，fixed，relative	检索对象的定位方式
top	auto，length	设置对象与其最近一个定位的父对象顶部相关的位置
right	auto，length	设置对象与其最近一个定位的父对象右边相关的位置
left	auto，length	设置对象与其最近一个定位的父对象左边相关的位置
bottom	auto，length	设置对象与其最近一个定位的父对象底边相关的位置
float	none,left,right	设置对象是否及如何浮动

position 属性各取值含义如下：

static：无特殊定位。

relative：定义该属性后，对象保持不动。如再设置 left，right，top，bottom 等属性，对象将按照这 4 个值在原来的位置上发生水平或垂直移动。

absolute：将对象从文档流中拖出，使用 left，right，top，bottom 等属性进行绝对定位。

z-index 属性各取值含义如下：

auto：遵循其父对象的定位。

length 自定义数值：无单位的整数值，可为负值。

top、right、bottom、left 属性各取值含义如下：

auto：无特殊定位。

length：自定义数值，由浮点数字和单位标识符组成的长度值，或百分数。必须定义 position 属性值为 absolute 或 relative 此取值方可生效。

float 属性各取值含义如下：

none：默认值。对象不飘浮。

left：文本流向对象的右边。

right：文本流向对象的左边。

跟随浮动对象的对象将移动到浮动对象的位置。浮动对象会向左或向右移动直到遇到边框、内边距、外边距或另一个块对象为止。

程序 3-7.html 演示了定位和 float 属性对页面排版的控制。

程序 3-7.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 3-7.html:边距属性-->
<head>
        <title>3-7.html</title>
        <style type="text/css">
       img{ border:1px gray rige;
           margin:5px 5px 5px 5px;
           padding:5px;
           float:left;
           width:100px;
           height:80px;
           }
       .img2{position:relative;
           top:-5px;
           right:10px;
           border:1px gray solid;
           margin:10px 10px 10px 0;
           padding:5px;
           float:right;
           width:130px;
           height:110px;
           }
       p{ margin:5px;
         padding:5px;
         font-size:13px;
         line-height:1.5;
         text-indent:2em;
```

```
            }
    div{width:500px;height:255px;border-style:solid;border-color:blue;
      border-width:1px;}
</style>
 </head>
<body>
<h3>埃德斯加·狄克斯特拉</h3>
 <div >
   <img src="Dijkstra2.jpg"/>
 <p>1972 年的图灵奖授予了荷兰的计算机科学家埃德斯加·狄克斯特拉
(Edsgar Wybe Dijkstra)。他对计算机科学作出了杰出贡献。他首个提出“goto 有害论”,
首创结构化程序设计，提出信号量和 PV 原语，解决了有趣的“哲学家聚餐”问题，创造了
Dijkstra 最短路径算法，第一个 Algol 60 编译器的设计者</p><img class=img2
src="Dijkstra1.jpg" />和实现者，THE 操作系统的设计者和开发者,与 D. E. Knuth
并称为
我们这个时代最伟大的计算机科学家。
<p>在与癌症进行了多年的斗争之后，伟大的荷兰计算机科学家 Edsger Wybe Dijkstra
已经于 2002 年 8 月 6 日在荷兰 Nuenen 自己的家中与世长辞！终年 72 岁。</p>
</div>
</body>
</html>
```

程序运行结果如图 3-8 所示。

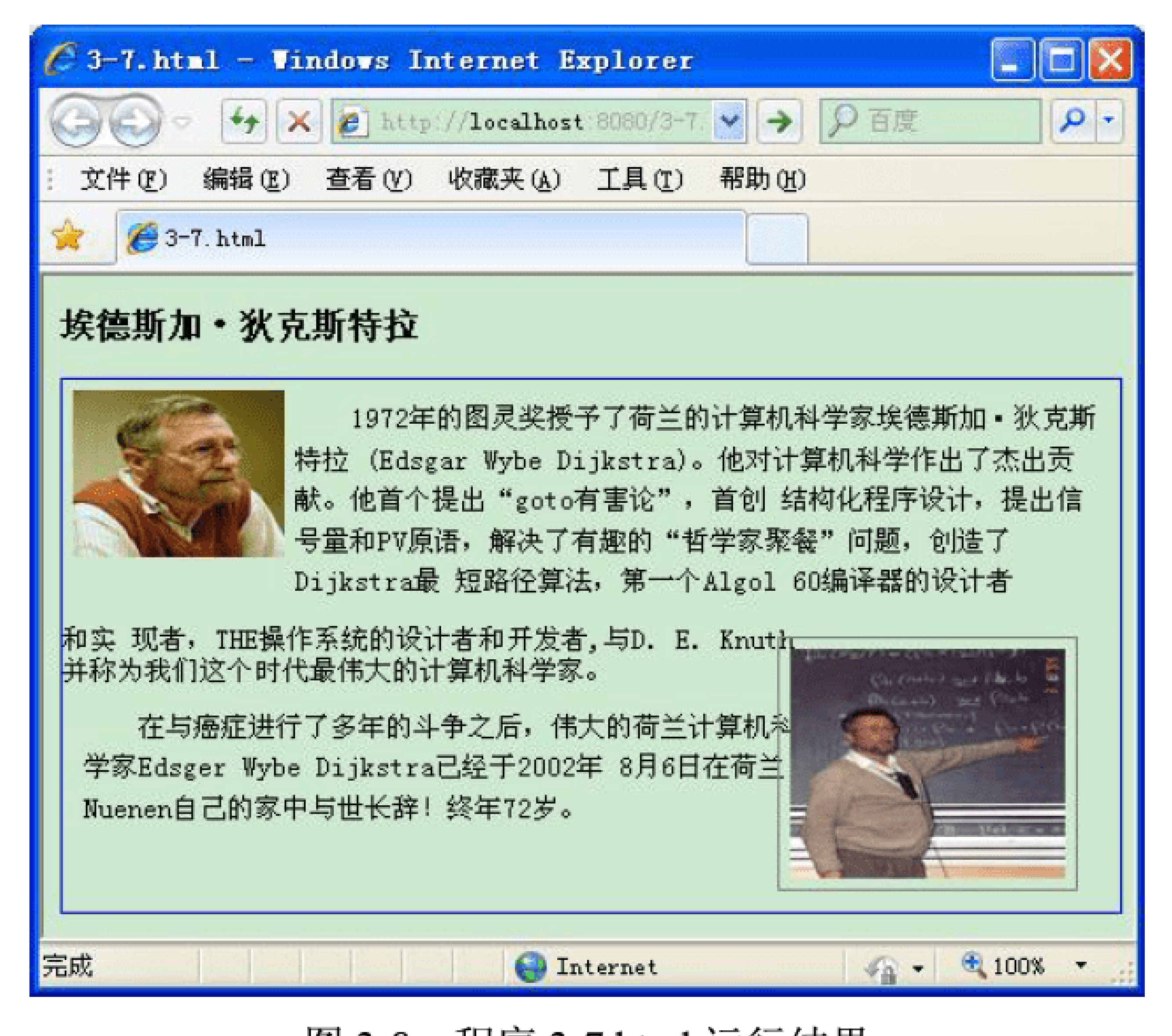


图 3-8 程序 3-7.html 运行结果

3.2.8 列表属性

列表属性允许放置、改变列表项标志，或将图像作为列表项标志。不是描述性的文本的任何内容都可以认为是列表。人口普查、太阳系、家谱、参观菜单，甚至所有朋友都可以表示为一个列表或者是列表的列表。控制列表的属性如表 3-9 所示。

表 3-9 列表属性

属 性	属 性 值	说 明
list-style-image	none，url(url)	设置列表项标记的图像
list-style-position	outside，inside	设置列表项标记如何根据文本排列
list-style-type	disc，circle，square，decimal，none	设置列表项所使用的预设标记

简写方式：

```
list-style : list-style-image || list-style-position || list-style-type
```

列表属性有个很有用的地方，就是制作横向菜单。程序 3-8.html 演示了用 list-style-type 属性制作横向菜单的方法。

程序 3-8.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <!--程序 3-8.html:用列表制作 CSS 横向菜单-->
<head>
<title>3-8.html</title>
  <style type="text/css">
#nav li {
            display: inline;
            list-style-type: none;
            padding: 5px;
        }
       a{
       text-decoration:none;
        }
     .div{
         width:400px;height:50px;
         background-color:yellow;
         padding-top:10px;
         }
  </style>
</head>
<body>
<div class=div>
<ul id="nav">
  <li><a href="#">我的首页</a></li>
  <li><a href="#">发表论文</a></li>
  <li><a href="#">毕业学生</a></li>
  <li><a href="#">研究方向</a></li>
  <li><a href="#">给我留言</a></li>
```

```
 </ul>
  </div>
</body>
</html>
```

程序运行结果如图 3-9 所示。



图 3-9　程序 3-8.html 运行结果

3.2.9　使用伪类

伪类是当某一事件发生时，才能发挥作用的样式。伪类不针对已经存在的静态目标元素。常用的伪类有 5 个：link（链接）、visited（访问过的链接）、hover（鼠标移动到相关元素上）、focus（获得焦点）和 active（对象激活）。其中前两种只能用于<a>标签，后 3 种为动态伪类，理论上可用于任何元素。

下面程序对程序 3-8.html 增加伪类的使用，就可以作出网页中菜单控制效果了。

程序 3-9.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <!--程序 3-9.html:使用伪类-->
<head>
<title>3-9.html</title>
  <style type="text/css">
#nav li {
            display: inline;
            list-style-type: none;
            padding: 5px;
        }

     .div{
         width:700px;height:80px;
         background-color:yellow;
         padding-top:10px;
         }
a:link{
         font-size:18px;color:black;text-decoration:none}
         a:hover{font-size:28px;color:red;text-decoration:none}
         a:active{font-size:18px;color:gray;text-decoration:none}
     a:visited{font-size:28px;color:blue;text-decoration:underline}
  </style>
</head>
<body>
<div class=div>
<ul id="nav">
  <li><a href="#">我的首页</a></li>
  <li><a href="#">发表论文</a></li>
  <li><a href="#">毕业学生</a></li>
  <li><a href="#">研究方向</a></li>
```

```
  <li><a href="#">给我留言</a></li>
</ul>
  </div>
</body>
</html>
```

程序运行结果如图 3-10 所示。

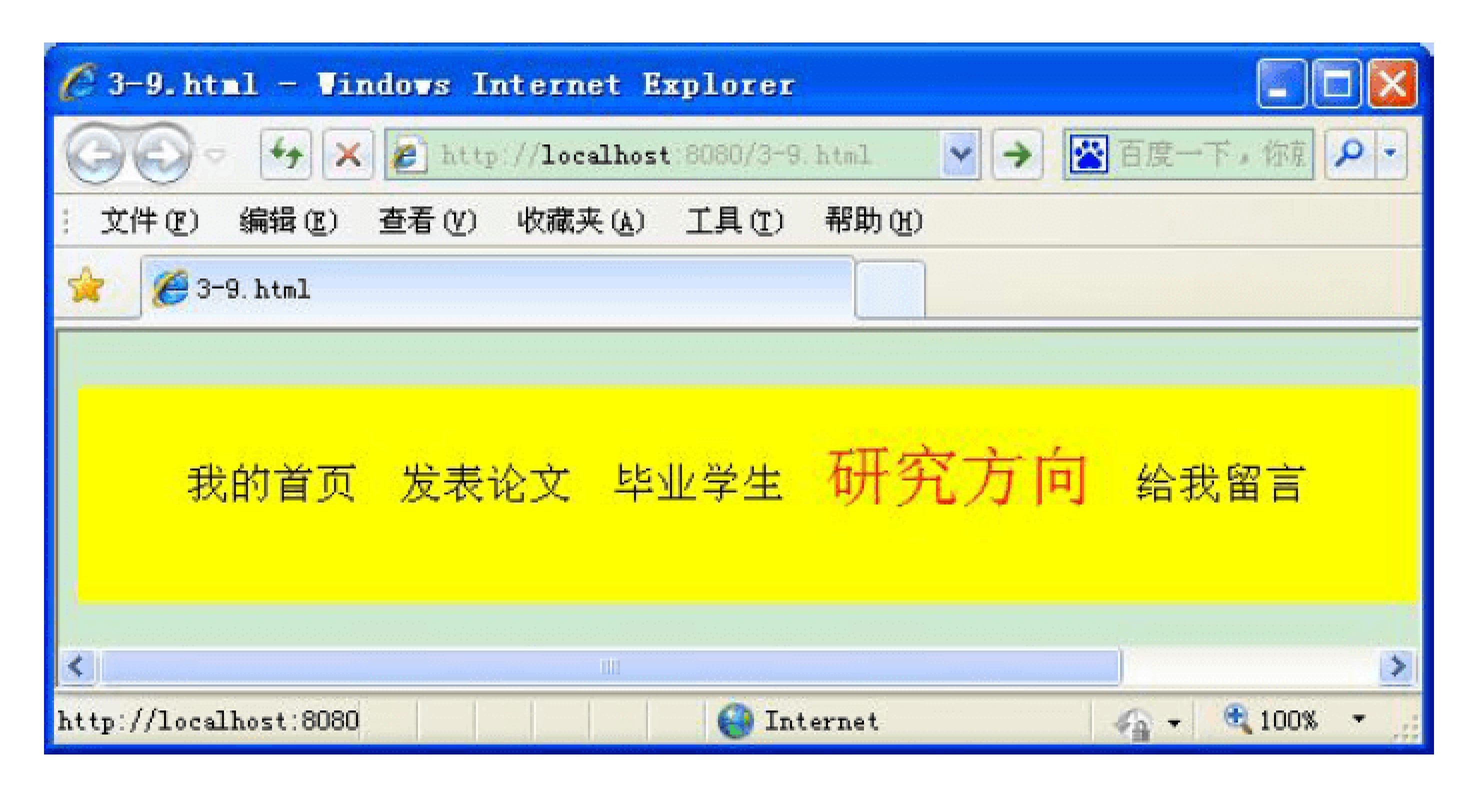


图 3-10 程序 3-9.html 运行结果

3.3 CSS+DIV 布局

页面布局一般有 3 种方式：表格布局、框架布局和 CSS+DIV 布局。其中，CSS+DIV 布局符合 W3C 标准，目前已成为网页布局主流。

<div>标签的主要作用是用于设定文字、图片、表格等的摆放位置。当把文字、图片等放在<div>标签中时，该标签被称为“DIV 块”或“DIV 元素”或“DIV 层”。

使用 CSS 和 DIV 可以很好地解决图像或文字定位的难题，通过 DIV 和 CSS 结合使用，网页设计人员可以精确地设定内容的位置，还可以将定位的内容上下叠放。使用 CSS+DIV 布局，可先将页面内容的语义或结构确定下来而不是先考虑外观。一个结构良好的 XHTML 页面可以通过 CSS 以任意外观表现出来。CSS 布局能实现真正意义上的结构和外观的分离，与传统的 TABLE 网页布局相比，具有以下 4 个显著优势：

（1）表现和内容相分离。将设计部分剥离出来放在一个独立样式文件中，XHTML 文件中只存放文本信息。

（2）提高搜索引擎对网页的索引效率。用只包含结构化内容的 XHTML 代替嵌套的标签，搜索引擎将更有效地搜索到你的网页内容，并可能给你一个较高的评价。

（3）提高页面浏览速度。对于同一个页面视觉效果，采用 CSS+DIV 重构的页面容量要比 TABLE 编码的页面文件容量小得多，前者一般只有后者的 1/2 大小。给用户最直观的体验就是网页打开速度快了很多，能给用户更好的体验。

（4）易于维护和改版。只要简单地修改几个 CSS 文件就可以重新设计整个网站的页面。

一种典型的网页布局如图 3-11 所示。

图中每个色块都是一个<div>，这里直接用 id 表示各个块。页面的所有 DIV 块都属于块#container，便于对页面的整体进行布局。

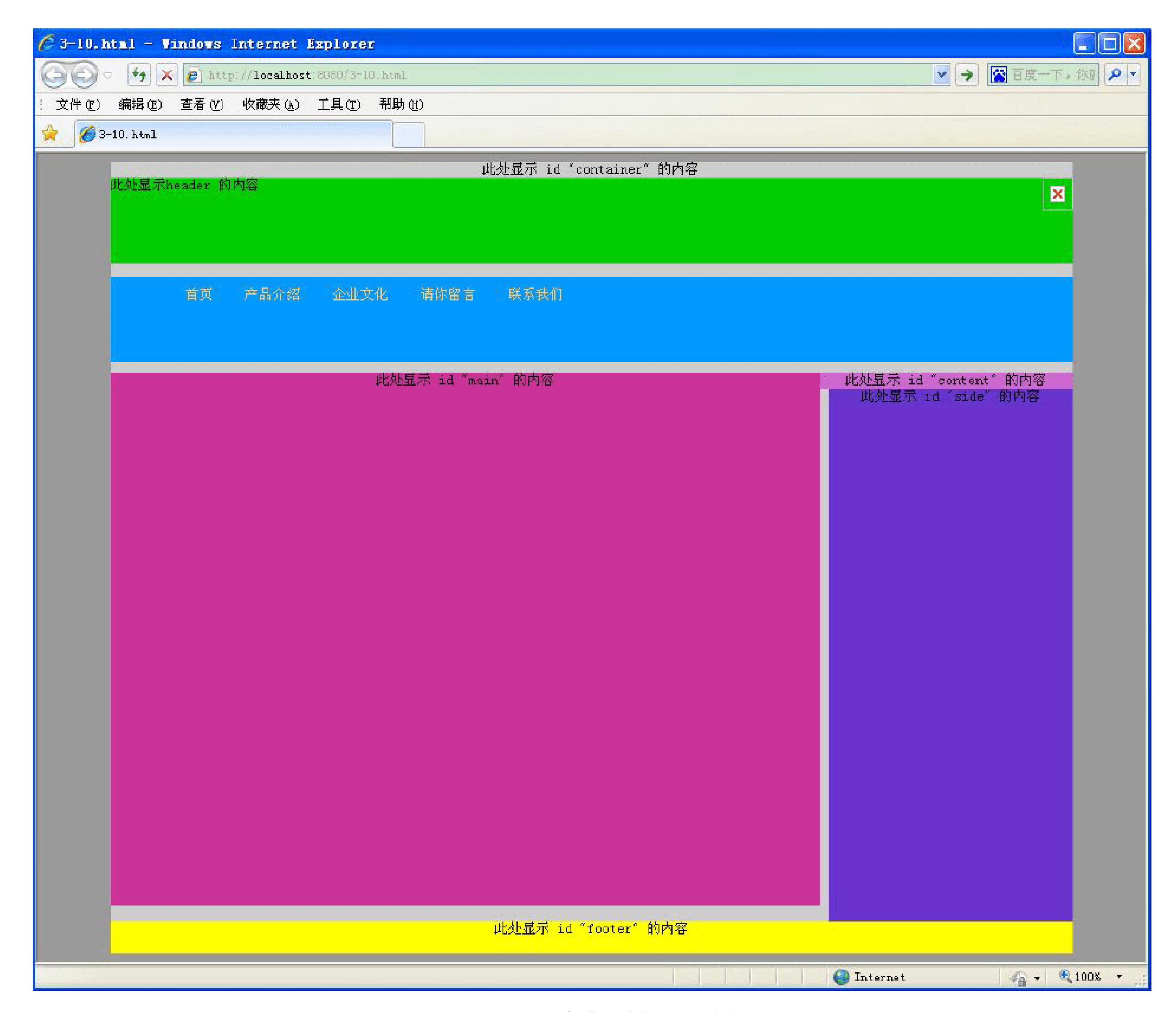


图 3-11 一种典型的网页布局

在每个子块内部，还要加入各种块元素或行内元素，构成网页。网页程序代码如下。

程序 3-10.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <!--程序 3-10.html:CSS+DIV 布局-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>3-10.html</title>
<style type="text/css">
body{ background:#999}
#container{background:#CCC; text-align:center; width:900px; margin:auto}
#header{ background:#0C0;  height:80px; margin-bottom:8px;}
#content{background:#C6C;}
#main{ background:#C39; width:664px; height:500px; float:left}
#side{background:#63C; width:228px; height:500px; float:right}
#footer{background:#FF0; height:30px}
```

```
.clearfloat {clear:both;height:0;font-size: 1px;line-height: 0px;}
.nav{width:900px; height:80px; text-align:center; margin:0px 0px 10px 0px; background:#09F}
.nav ul{list-style:none}
.nav li{float:left; padding-left:30px; padding-top:10px}
.nav a:link{ color:#FC6; text-decoration:none}
.nav li a:visited{ color:#FFF; text-decoration:none}
.nav li a:hover{ background:#C30; padding:5px; }
.nav li a:active{}
#header img{ float:right;}
#header span{float:left}
</style>
</head>
<body>
<div id="container">此处显示  id "container" 的内容
  <div id="header">
  <div ><span>此处显示 header 的内容</span></div>
  <div><img src="images/1.png" /></div>
 </div>
  <div class="nav">
<ul>
<li ><a href="#">首页</a></li>
<li><a href="#" onclick="hhh">产品介绍</a></li>
<li><a href="#">企业文化</a></li>
<li><a href="#">请你留言</a></li>
<li><a href="#">联系我们</a></li>
</ul>
  </div>
  <div id="content">此处显示  id "content" 的内容
    <div id="main">此处显示  id "main" 的内容</div>
    <div id="side">此处显示  id "side" 的内容</div>
  </div>
  <div class="clearfloat"></div>
  <div id="footer">此处显示  id "footer" 的内容</div>
</div>
</body>
</html>
```

3.4　本章小结

网页的组成分为 3 个部分：结构、表现和行为，CSS 就是控制网页表现的标准。CSS 是一组格式设置规则，表能实现网页内容与表现形式的分离，方便团队开发。使用 CSS 能弥补 XHTML 对网页格式化功能的不足，如段落间距、行距等，能灵活地控制字体变化和大小，方便页面格式的动态更新，并能对页面进行精准地排版定位等。DIV 是组成网页的

一种块级元素，利用 DIV 和 CSS 的定位技术进行网页的布局是现在主流的页面布局模式，是 TABLE 页面布局的替代技术。采用 CSS+DIV 布局的页面容量要比用 TABLE 布局的页面文件小很多，前者一般只有后者的 1/2 大小，节省了大量的带宽，同时降低了网站改版的成本。W3C 组织提供对 CSS 代码的验证服务，地址为 http://jigsaw.w3.org/css-validator/。

3.5 练习题

1．什么是 CSS 和 DIV？
2．CSS 嵌入 XHTML 的方法有哪些？
3．CSS 的选择符分几类？分别举例说明。
4．如何在网页中设置字体？如何设置列表属性？
5．什么是伪类？其主要用途是什么？
6．什么是相对定位？什么是绝对定位？
7．CSS+DIV 布局的优点有哪些？缺点又是什么？
8．编程实现纯 CSS 制作的下拉菜单。
9．编程实现用 CSS 控制表格的显示。
10．编程实现一个个人主页，用 CSS+DIV 布局，用 CSS 控制页面显示样式和风格。

第4章 客户端交互——JavaScript

本章重点

- JavaScript 的基础语法;
- JavaScript 内置对象;
- 浏览器对象模型;
- JavaScript 的事件处理。

从以上可知，进行 Web 开发要遵循一定的标准，Web 标准是一系列标准的集合。网页主要由 3 部分组成：结构（Structure）、表现（Presentation）和行为（Behavior）。第 2 和第 3 章分别介绍了结构和表现的控制方法，本章就要介绍网页上的行为控制了。

XHTML 和 CSS+DIV 的工作是设计和规划页面的内容和显示效果，但所提供的页面是静态的，缺少交互和动态效果，只是一个针对人阅读的发布平台。JavaScript 弥补了以上不足，大大增强了客户端 Web 页面的动态性和交互性，极大地提升了用户的上网体验，是目前客户端开发的事实上的标准。

JavaScript 是由 Netscape 公司开发的一种脚本语言。JavaScript 自然可以做很多的事情，不过它最主流的应用还是在 Web 上——创建动态网页，几乎所有的动态网页里都能找到它的身影。目前流行的 Ajax 也是依赖于 JavaScript 而存在的。

JavaScript 由 3 个部分组成：核心、客户端和服务端。服务端应用较少，本书的服务端使用 PHP。因此，本书中不介绍 JavaScript 的服务端。客户端的 JavaScript 可以完成一些本来该由服务端完成的任务，但不能取代所有的服务器计算，如服务端的文件操作、数据库访问和网络连接等功能，客户端的 JavaScript 都不支持。

4.1 JavaScript 的特点

概括地说，JavaScript 是一种基于对象和事件驱动，并具有安全性能的脚本语言。对象和事件是 JavaScript 的两个核心。

使用 JavaScript 可以将动态的文本放入 XHTML 页面，可以对网页上发生的事件作出响应，可以读写 XHTML 元素的内容，在数据被提交到服务器之前，可被用来验证这些数

据，用来检测访问者的浏览器并为检测到的浏览器载入相应的页面。JavaScript 用来创建 Cookies，存储和取回位于访问者的计算机中的信息。

JavaScript 是一种脚本语言。脚本语言的特点是比较简单、易学，即使是程序设计新手也可以非常容易地使用 JavaScript 进行简单的编程。

4.2 将 JavaScript 插入网页的方法

在网页中插入 JavaScript 语言的方法有 3 种：

（1）使用<script>标记的 language 属性将脚本嵌入到网页中；

（2）将脚本嵌入到 XHTML 标记的事件中；

（3）通过<script>标记的 src 属性链接外部脚本文件。

下面对这 3 种方法分别介绍。

4.2.1 使用<script>标记的 language 属性

使用下面的代码可以在网页中插入 JavaScript：

```
<script type="text/JavaScript" language="javascript">
    JavaScript 代码
</script>
```

其中，language="javascript"表示使用 JavaScript 脚本语言，脚本语言还有 vbscript、jsscript 等，如果没有 language 属性，表示默认使用 JavaScript 脚本。例如：

```
<script type="text/JavaScript">
  document.write("Hello,World!");
</script>
```

JavaScript 使用 document.write 输出内容。上面的代码在网页上输出“Hello,World!”。个别浏览器可能不支持 JavaScript，需要使用如下方法对它们隐藏 JavaScript 代码：

```
<html>
  <body>
  <script type="text/JavaScript">
  <!--
  document.write("Hello,World!");
  //-->
  </script>
  </body>
</html>
```

<!-- -->中的内容对于不支持 JavaScript 的浏览器来说就等同于一段注释，而对于支持 JavaScript 的浏览器，这段代码仍然会执行。至于“//”符号则是 JavaScript 里的注释符号，在这里添加它是为了防止 JavaScript 试图执行-->。通常情况下，现在的浏览器几乎都支持 JavaScript，即使是不支持的，也会了解如何合理地处理含有 JavaScript 的网页。JavaScript 的注释约定为，单行的注释以 // 开始，多行注释以/*开头，以*/结尾。

4.2.2　直接嵌入到 XHTML 标记的事件中

可以直接在 XHTML 的某些标记内添加事件，然后将 JavaScript 脚本写在该事件的值内，以输入相应元素的事件。例如：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head></head>
  <body>
  <img src="11.jpg" onClick="JavaScript:alert('Hello,World!');"/>
  </body>
</html>
```

代码在<img/>标记的 onClick 事件中嵌入了一小段 JavaScript 代码。

4.2.3　通过<script>标记的 src 属性链接外部脚本

将 JavaScript 源代码保存成扩展名为 js 的外部文件，然后可以通过<script>标记的 src 属性将其嵌入到引用它的 XHTML 文件，例如：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <body>
  <script language="JavaScript" type="text/JavaScript" src="myjs.js">
  </body>
</html>
```

在代码中，通过 src 属性，链接外部脚本文件，对保存在外部文件中的 JavaScript 脚本进行了引用。

4.3　插入 JavaScript 的位置

JavaScript 脚本可以放在网页的 head 里或者 body 部分，而且效果也不相同。

（1）放在 body 部分的 JavaScript 脚本在网页读取到该语句的时候就会执行，但放在 body 中的函数被调用时才会执行。

（2）在 head 部分的脚本在被调用时才会执行，通常是在<script>…</script>定义函数，通过调用函数执行 head 里的脚本。

注意：脚本文件里头不能再含<script>标签。

4.4　JavaScript 语言基础

4.4.1　JavaScript 数据类型和变量

JavaScript 支持数值型、字符串型、布尔型、空类型和未定义类型 5 种基本数据类型和

数组、对象两种复合数据类型。数值型数据不区分整型和浮点型，英文状态下的单引号或双引号中的部分就是一个字符串，布尔型数据的值是 true 或 false，空类型只有一个值，就是 null，表示没有任何值。未定义类型也只有一个值，即 undefined。JavaScript 是一种弱类型的语言，存放一个数据的变量或常量不必首先作声明，而是在使用或赋值时确定其数据的类型。在 JavaScript 中任何变量都是用关键字 var 定义的。如下代码声明了 4 个变量：

```
<script language="javascript" type="text/jscript">
  var a;         //声明一个变量
  var b="hello";//声明一个变量并初始化
  var c,d;       //同时声明了两个变量
</script>
```

变量的命名要遵守：

（1）首字符必须是字母或下划线或$。

（2）首字符后面的字符可以是字母、数字、下划线或$。

（3）变量名中不能包含空格、回车符或其他标点字符。

（4）变量名对大小写敏感。

（5）不能使用 JavaScript 的关键字或保留字。

在 JavaScript 的变量中同样有全局变量和局部变量。全局变量定义在所有函数体之外，对任何函数可见。局部变量定义在函数体之内，只对该函数可见。

4.4.2 JavaScript 保留字和转义字符

保留字不能作为变量名和函数名使用。表 4-1 所示为当前 ECMA 262 规范中的保留字。

表 4-1 JavaScript 的保留字

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	finally	instanceof	throw	while
default	for	new	try	with

为了适应发展变化，还预留了如表 4-2 所示的单词，因此它们也不应该用于程序中。

表 4-2 JavaScript 预留的关键字

abstract	double	implements	private	throws
boolean	enum	import	protected	transient
byte	export	int	public	volatile
char	extends	interface	short	
class	final	long	static	
const	float	native	super	
debugger	goto	package	synchronized	

有些字符不能直接写在引号中。如果字符串要使用这些特殊字符，就必须利用转义字符表示，JavaScript 中的转义字符如表 4-3 所示。

表 4-3　JavaScript 中的转义字符

字 面 量	含 义	字 面 量	含 义
\n	换行	\'	单引号
\t	制表符	\"	双引号
\b	空格	\0nnn	八进制
\r	回车	\xnn	十六进制
\f	换页符	\unnn	十六进制 Unicode 字符
\\	反斜杠		

4.4.3　JavaScript 的运算符和表达式

变量用来存放数据，运算符则用来处理数据。用运算符将变量和常量连接起来则组成表达式。JavaScript 中的运算符有如下 7 类。

赋值运算符：=，+=，–=，*=，/=，%=，<<=，>>=，|=，&=。

算术运算符：+，–，*，/，++，– –，%。

比较运算符：>，<，<=，>=，==，===，!=，!==。

逻辑运算符：||，&&，!。

条件运算：? :。

位移运算符：|，&，<<，>>，~，^。

字符串运算符：+。

其中，===为完全相等，!==为不完全相等。其他运算符的含义与 C 语言中的定义相同。

用运算符将操作数连接起来，就形成了表达式。表达式再加上“;”号，就形成了语句。

4.4.4　JavaScript 的语句

同其他编程语言一样，JavaScript 也是通过语句实现程序的流程控制的，包括条件语句、循环语句、for…in 语句和 with 语句。本节主要介绍 JavaScript 中不同于其他编程语言的语句。

1. for…in 语句

这是 JavaScript 提供的一种特别的循环方式，用来遍历一个对象的所有用户定义的属性或一个数组的所有元素。for…in 循环中的计数器是一个字符串而不是数字，包含了当前对象属性的名称或表示当前数组元素的下标。其语法格式如下：

```
for(变量 in 对象或数组){循环体语句}
```

循环体语句每次执行前，将 in 后面所遍历的数组的下一个元素下标或对象的下一个属性名赋值给变量。在循环体语句中，变量可以直接调用。在有些情况下，开发者无法预知对象的任何信息，更谈不上控制循环次数，这时用 for…in 语句可以很好地解决问题。

例如，用 for…in 遍历数组：

```
<script language="javascript">
  var arr=new Array(1,2,3,4,5,6);
  for(i in arr)
  { document.write("数组元素 arr["+i+"]为"+arr[i]);
    document.write("<br>");
}
</script>
```

输出结果为：

```
数组元素 arr[0]为 1
数组元素 arr[1]为 2
数组元素 arr[2]为 3
数组元素 arr[3]为 4
数组元素 arr[4]为 5
数组元素 arr[5]为 6
```

2. with 语句

当需要访问某个对象的若干属性和方法时，通常要在属性和方法名前加上该对象名，书写较为烦琐，使用 with 语句可以简化对对象的属性和方法的访问。with 语句的作用是：在该语句体内，任何对变量的引用被认为是对这个对象的属性的引用。其格式如下：

```
with(对象名){  段语句}
```

例如：

```
<script language="javascript">
  date=new Date();
  with (date)
  { year=getYear();
    month=getMonth();
  day=getDay();
  }
</script>
```

以上代码等价于

```
<script language="javascript">
  date=new Date();
  year=date.getYear();
  month=date.getMonth();
  day=date.getDay();

</script>
```

4.4.5 JavaScript 的函数

函数要先定义，后调用。通常在<head>标记中定义函数，在<body>标记中调用定义好的函数。在 JavaScript 中，定义函数要使用关键字 function，其基本语法如下：

```
function functionName(参数 1，参数 2，…,参数 n)
```

```
{ 函数体语句段
  return [表达式]
}
```

函数定义后，就可以调用该函数完成所定义的操作。

4.4.6　第一个 JavaScript 程序

以下用一个最为简单的例子程序 4-1 来演示 JavaScript 程序的写法。

程序 4-1

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <!--程序 4-1.html:第一个 JavaScript 程序-->
05 <head>
06 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
07 <title>4-1.html</title>
08 <script language="javascript">
09   function add(a,b){return a+b;}
10 </script>
11 </head>
12 <body>
13 <script language="javascript">
14   document.writeln("3+2="+add(3,2));
15 </script>
16 </body>
17  </html>
```

其中，JavaScript 是嵌入到 html 文件中的。程序的第 8 行和第 13 行分别是嵌入 JavaScript 代码的位置，第 9 行定义了一个名为 add 的函数，第 14 行调用了这个函数，函数执行的结果用 document.writeln 输出到屏幕上。程序运行结果如图 4-1 所示。



图 4-1　程序 4-1 运行结果

4.5　JavaScript 内置对象

与面向对象的语言不同，JavaScript 是基于对象的语言，其中包含了一些已经创建好的对象。程序员使用这些对象而不用创建新的类。因此 JavaScritp 中没有提供定义类的关键字 class。Object 对象是所有内置对象的根对象，提供了 JavaScript 对象的最基本功能，这些功能构成了其他所有 JavaScript 对象的基础。直接使用 Object 对象的情况比较少，更多地是使用其他如 String、Date、Math、Number、Array 等对象。

4.5.1　String 对象

String 对象提供了对字符串的操作功能，需要定义对象的实例后才能应用它的属性和

方法，使用点“.”调用 String 对象的属性和方法。

创建字符串对象的方法有两种：一种是通过用单引号或双引号引起来的字符串赋值给变量，称作隐式方法；另一种是使用关键字 new 和字符串对象构造函数 String 创建 String 对象。示例如下：

```
var string1="Very Well";
var string2=new String("Very Well");
var string3=new String('Very Well');
var string4=new String(mystring1);
```

String 对象的属性只有一个 length 属性。它用于获得字符串的长度。语法格式：

字符串变量名.length

例如：

```
var s = "JavaScript";
var strlen = s.length;          //strlen 为 10
```

该实例代码返回字符串"JavaScript"的长度 10。

length 属性会随着字符串的变化自动更新，开发人员不能对该属性进行直接设置。

String 对象的方法非常丰富，可分为两类：一类方法模拟 XHTML 标记，用于格式化字符串的显示，如改变字体大小、文字颜色等；另一类方法用于操作处理字符串，如查找和替换字符串中的字符、改变字符串的大小写、提取字符串中的子串等。其常用方法如表 4-4 所示。

表 4-4　String 对象的常用方法

方　法　名	功　　能
Big	为字符添加大体 XHTML 标记
Small	为字符添加小体 XHTML 标记
Italics	为字符添加斜体 XHTML 标记
Bold	为字符添加粗体 XHTML 标记
Blink	为字符添加闪烁 XHTML 标记
Fixed	为字符添加固定高亮 XHTML 标记
Fontsize	为字符添加控制字体大小 XHTML 标记
Tolowercase	将字符串转换为小写
Touppercase	将字符串转换为大写
Indexof	从指定字符位置开始搜索某一特定字符第一次出现的位置
Substring	返回所设置的一部分字符串

4.5.2　Date 对象

程序设计中经常需要处理时间、日期，JavaScript 中 Date 对象用于日期和时间的处理。创建 Date 对象的语法：

```
var myDate=new Date();
```

Date 对象会自动把当前日期和时间保存为其初始值。

参数形式有以下 5 种：

```
new Date("month dd,yyyy hh:mm:ss");
new Date("month dd,yyyy");
new Date(yyyy,mth,dd,hh,mm,ss);
new Date(yyyy,mth,dd);
new Date(ms);
```

注意：最后一种形式，参数表示的是需要创建的时间和 GMT 时间 1970 年 1 月 1 日之间相差的毫秒数。各种函数的含义如下：

- month：用英文表示月份名称，从 January 到 December。
- mth：用整数表示月份，从 0（1 月）到 11（12 月）。
- dd：表示一个月中的第几天，从 1～31。
- yyyy：4 位数表示的年份。
- hh：小时数，从 0（午夜）～23（晚 11 点）。
- mm：分钟数，从 0～59 的整数。
- ss：秒数，从 0～59 的整数。
- ms：毫秒数，为大于等于 0 的整数。

例如：

```
new Date("January 12,2006 22:19:35");
new Date("January 12,2006");
new Date(2006,0,12,22,19,35);
new Date(2006,0,12);
new Date(1137075575000);
```

Date()返回当日的日期和时间。

getDate()从 Date 对象返回一个月中的某一天（1～31）。

getDay()从 Date 对象返回一周中的某一天（0～6）。

GetMonth()从 Date 对象返回月份（0～11）。

getFullYear()从 Date 对象以 4 位数字返回年份。

getYear()请使用 getFullYear()方法代替。

getHours()返回 Date 对象的小时（0～23）。

getMinutes()返回 Date 对象的分钟（0～59）。

getSeconds()返回 Date 对象的秒数（0～59）。

getMilliseconds()返回 Date 对象的毫秒（0～999）。

getTime()返回 1970 年 1 月 1 日至今的毫秒数。

下面用例子说明 Date 对象的使用。程序 4-2 演示了用 Date 对象处理时间日期的方法。

程序 4-2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 4-1.html:用 Date 对象处理时间日期-->
```

```
<html>
  <head>
   <script type="text/javascript">
    function startTime()
    {
    var today=new Date()
    var h=today.getHours()
    var m=today.getMinutes()
    var s=today.getSeconds()
    // add a zero in front of numbers<10
    m=checkTime(m)
    s=checkTime(s)
    document.getElementById('txt').innerHTML=h+":"+m+":"+s
    t=setTimeout('startTime()',500)
    }

    function checkTime(i)
    {
    if (i<10)
      {i="0" + i}
      return i
     }
   </script>
   <title>4-2.html</title>
  </head>
    <body onload="startTime()">
    <div id="txt"></div>
    </body>
</html>
```

程序运行结果如图 4-2 所示。



图 4-2 程序 4-2 运行结果

4.5.3 Math 对象

Math 对象提供了一组在进行数学运算时非常有用的属性和方法。Math 对象的属性是一些常用的数学常数，如表 4-5 所示。

表 4-5 Math 对象的属性

Math 属性	说 明
E	自然对数的底
LN2	2 的自然对数
LN10	10 的自然对数
LOG2E	底数为 2，真数为 E 的对数
LOG10E	底数为 10，真数为 E 的对数
PI	圆周率的值
SORT1_2	0.5 的平方根
SORT2	2 的平方根

Math 对象的方法可直接引用，实现数学计算，常用的方法及说明如表 4-6 所示。

表 4-6　Math 对象的常用方法

Math 方法	说　　明
sin(x)/cos(x)/tan(x)	返回 x 的正弦/余弦/正切值
asin(x)/acos(x)/atan(x)	返回 x 的反正弦/反余弦/反正切值
abs(x)	返回 x 的绝对值
ceil(x)	返回大于等于 x 的最小整数
floor(x)	返回小于等于 x 的最大整数
exp(x)	返回 E（自然对数的底）的 x 次幂
log(x)	返回 x 的自然对数
pow(x)	返回 x 的指定次幂
random()	返回一个[0，1）之间的随机小数
sqrt(x)	返回 x 的平方根

4.5.4　Array 对象

数组（Array）是常见的一种数据结构，可以用来存储一系列的数据。数组中提供了 4 个属性，最常用的是数组大小属性 length。JavaScript 中用 Array 对象表示数组，创建数组的方式如下面代码所示：

```
// 不带参数，返回空数组。length 属性值为 0
new Array();
// 数字参数，返回大小为 size 的数组。length 值为 size，数组中的
// 所有元素初始化为 undefined
new Array(size);
// 带多个参数，返回长度为参数个数的数组。length 值为参数的个数
new Array(e1, e2, ..., eN);
```

数组中常用的方法如表 4-7 所示。

表 4-7　Array 对象的常用方法

Array 方法	说　　明
concat	连接两个或更多的数组，并返回合并后的新数组
join	把数组的所有元素放入一个字符串并返回此字符串。元素通过指定的分隔符进行分隔
pop	删除并返回数组的最后一个元素
push	向数组的末尾添加一个或更多元素，并返回新的长度
reverse	颠倒数组中元素的顺序
sort	对数组的元素进行排序
tostring	把数组转换为字符串，并返回结果

程序 4-3 演示了 array 对象的应用。

程序 4-3

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 4-3.html:用 Array 对象处理数据-->
<head>
  <title>4-3.html</title>
</head>
<body>
<script language="JavaScript" type="text/javascript">
  var array=[12,11,15,24,56,25,67,33,54,98];
  for(var i=0;i<array.length;i++){
   document.write(array[i]+",");
  }
  document.write("<br/><br/>");
  function change(array){
    for(var i=0;i<array.length;i++){
      for(var j=0;j<array.length;j++){
        if(array[i]<array[j]){
          var temp=array[i];
          array[i]=array[j];
          array[j]=temp;
        }
      }
    }
  }
 change(array);
 for(var i=0;i<array.length;i++){
   document.write(array[i]+",");
 }
 document.write("<br/><br/>");
 array.reverse();

 for(var i=0;i<array.length;i++){
   document.write(array[i]+",");
  }
</script>
</body>
</html>
```

程序运行结果如图 4-3 所示。

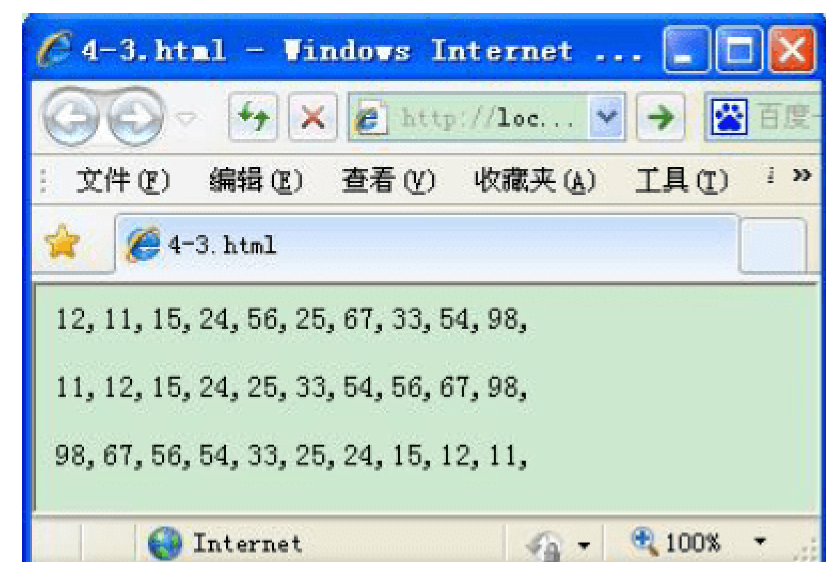


图 4-3 程序 4-3 运行结果

4.6 浏览器对象模型

JavaScript 提供了一系列对象用于与 JavaScript 运行环境即浏览器的交互。这些对象是 window、document、location、history、navigator 和 screen 等，这些对象间的层次关系

如图 4-4 所示。

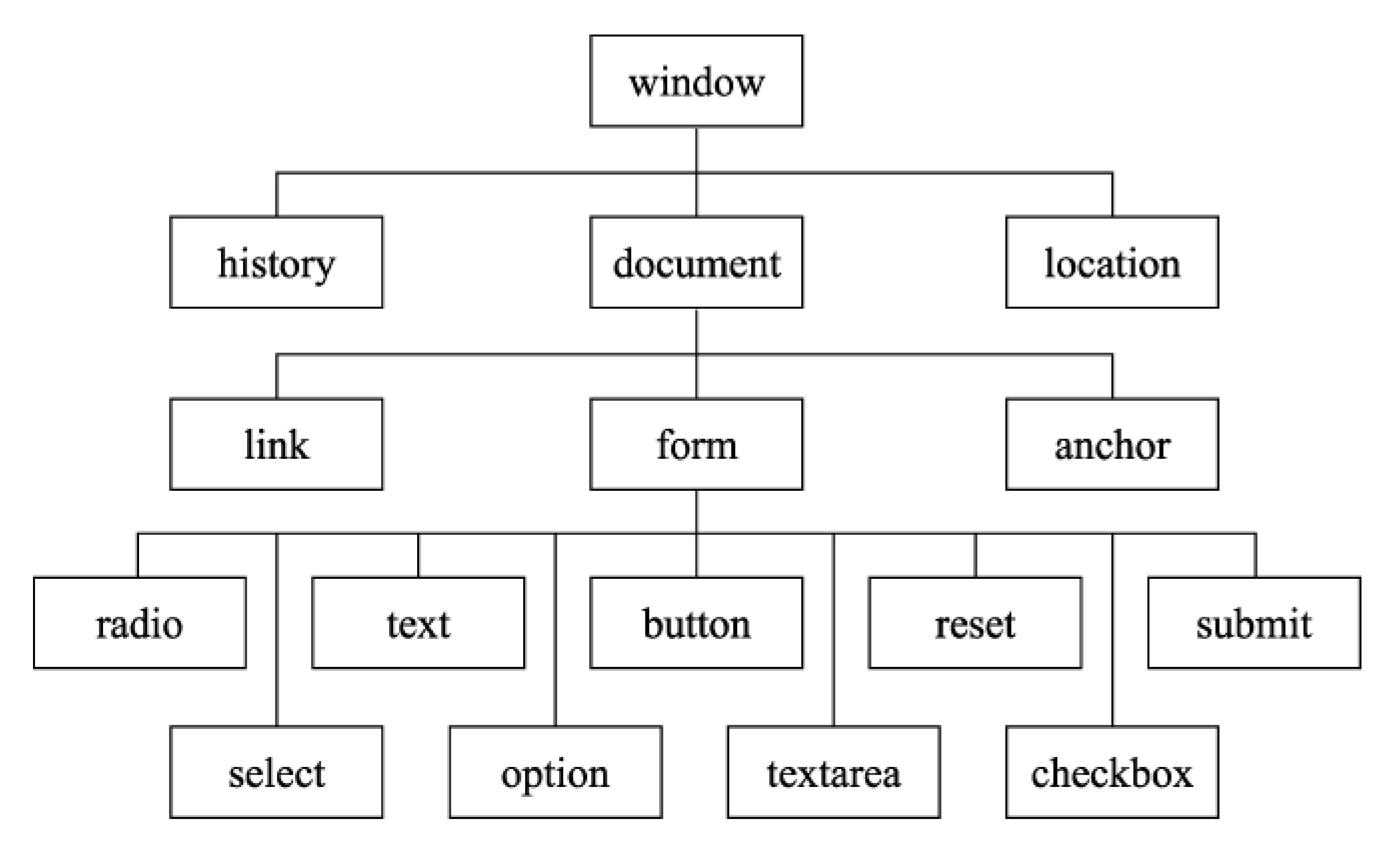


图 4-4　浏览器对象层次关系图

- window 对象在层次图中位于最高一层，document 对象、location 对象和 history 对象都是它的子对象，window 对象中包含的属性是应用于整个窗口的。
- document 对象在层次图中位于最核心的地位，页面上的对象都是 document 对象的子对象，在 document 对象中包含的属性是整个页面的属性，如背景颜色、标题、表单对象等。
- location 对象中包含了当前 URL 地址的信息。
- navigator 对象中包含了当前使用的浏览器的信息，包括客户端浏览器支持的 MIME 类型信息和所安装的插件信息。
- history 对象中包含了客户端浏览器过去访问的 URL 地址信息。

本节主要介绍应用最多的两个对象：window 对象和 document 对象。

4.6.1　window 对象

window 对象表示的是浏览器窗口，其他对象都继承自 window 对象，并且是通过 window 对象访问的。window 对象可用于检索有关窗口的状态信息，也可以用来显示 document 对象和访问窗口中发生的事件。window 是 JavaScript 的高级对象，通常在 JavaScript代码编写中应用它的方法、属性时，window对象并不是必须写出的。如引用window 的 parent 属性时，window.parent 可以简单写成 parent。window 对象中包含的属性是应用于整个窗口的。在浏览器中，window 对象是所有对象的根对象，只要打开了浏览器窗口，不管该窗口中是否有打开的网页，当遇到 body、frameset 或 frame 元素时，都会自动创建 window 对象的实例。

window 对象的主要属性如表 4-8 所示。

表 4-8　window 对象的主要属性

window 属性	说　明
name	可读写属性，表示当前窗口的名称
parent	只读属性，如果当前窗口有父窗口，表示当前窗口的父窗口对象

续表

window 属性	说　　明
opener	只读属性，表示产生当前窗口的窗口对象
self	只读属性，表示当前窗口对象
top	只读属性，表示最上层窗口对象
defaultstatus	可读写属性，表示在浏览器的状态栏中显示的缺省内容
status	可读写属性，表示在浏览器的状态栏中显示的内容
document	表示在窗口中显示的 HTML 文件
location	表示窗口中显示的文档的 URL

常用的 window 方法如表 4-9 所示。

表 4-9　window 对象的常用方法

window 方法	说　　明
alert	显示带有一段消息和一个确认按钮的警告框
clearInterval	取消由 setInterval()设置的计时器
clearTimeout	取消由 setTimeout()方法设置的计时器
close	关闭浏览器窗口
confirm	显示带有一段消息以及确认按钮和取消按钮的对话框
focus	把键盘焦点给予一个窗口
open	打开一个新的浏览器窗口或查找一个已命名的窗口
prompt	显示可提示用户输入的对话框
setInterval	按照指定的周期（以毫秒计）调用函数或计算表达式
setTimeout	在指定的毫秒数后调用函数或计算表达式

在这些方法中，open 方法使用较为频繁，该方法用以打开一个新窗口，语法格式如下：

```
window.open(url,name,features,replace)
```

open 方法中的 features 参数表示新建窗口的特征，该参数的取值如表 4-10 所示。

表 4-10　参数 features 的取值

features 值	说　　明
channelmode	是否使用 channel 模式显示窗口，默认为 no，可选值为 yes|no|1|0
directories	是否添加目录按钮。默认为 yes，可选值为 yes|no|1|0
fullscreen	是否使用全屏模式显示
height	文档显示区的高度，单位是像素
left	x 坐标，单位是像素
location	是否显示地址字段，默认是 yes，可选值为 yes|no|1|0

续表

features 值	说　　明
menubar	是否显示菜单栏，默认是 yes，可选值为 yes|no|1|0
resizable	是否可调节尺寸，默认是 yes，可选值为 yes|no|1|0
scrollbars	是否显示滚动条，默认是 yes，可选值为 yes|no|1|0
status	是否添加状态栏，默认是 yes，可选值为 yes|no|1|0
titlebar	是否显示标题栏，默认是 yes，可选值为 yes|no|1|0
toolbar	是否显示浏览器的工具栏，默认是 yes，可选值为 yes|no|1|0
top	y 坐标，单位是像素
width	文档显示区的宽度，单位是像素

4.6.2　document 对象

document 对象指的是在浏览器窗口中显示的 XHTML 文档，也是 window 对象的一个属性。document 对象提供了交互访问静态文档内容的功能，可以把静态 XHTML 文档转换成交互程序，是客户端使用最多的 JavaScript 对象。

document 对象的属性如表 4-11 所示。

表 4-11　document 对象的属性

属　　性	说　　明
bgColor	设置或获取表明对象后面的背景颜色的值
fgColor	设置或获取文档的前景（文本）颜色
linkColor	设置或获取对象文档链接的颜色
body	提供对<body>元素的直接访问。对于定义了框架集的文档，该属性引用最外层的<frameset>
cookie	设置或返回与当前文档有关的所有 cookie
domain	返回当前文档的域名
lastModified	返回文档被最后修改的日期和时间
referrer	返回载入当前文档的 URL
title	返回当前文档的标题
URL	返回当前文档的 URL
forms[]	返回文档中 form 的集合
imgas[]	返回文档中<img>比记的集合

document 对象的方法如表 4-12 所示。

表 4-12 document 对象的方法

方法名	说明
getElementById	返回对拥有指定 id 的第一个对象的引用
getElementsByName	返回带有指定名称的对象集合
getElementsByTagName	返回带有指定标签名的对象集合
write	向文档写 HTML 表达式或 JavaScript 代码
writeln	向文档写 HTML 表达式或 JavaScript 代码并换行

4.7 事件与事件处理

事件是用户和 Web 页面交互时产生的各种操作。例如，用户单击按钮或超链接时，就产生 click 事件，当输入框或文本域中输入的字符值改变时产生 change 事件。JavaScript 中所有事件都是对象，都是由小写字母组成的。浏览器运行的大部分时间都是在等待事件的发生，并在事件发生时调用相应的事件处理函数，完成事件处理。

发生事件的地方称为事件源，任何 XHTML 标记都可成为事件源。发生事件时调用的处理函数成为事件处理器，处理器的命名原则为 on+事件名，如 click 事件的处理器名为 onclick。表 4-13 所示为 JavaScript 中常见的事件。

表 4-13 JavaScript 中常见的事件

事件	标签	标签属性	说明
blur	<a>	onblur	链接失去输入焦点
	<button>		按钮失去输入焦点
	<input>		输入元素失去输入焦点
	<textarea>		文本域失去输入焦点
	<select>		选择元素失去输入焦点
change	<input>	onchange	输入元素被修改并失去输入焦点
	<textarea>		文本域被修改并失去输入焦点
	<select>		选择元素被修改并失去输入焦点
click	<a>	onclick	单击链接
	<input>		单击输入元素
dbclick	绝大部分元素	ondbclick	双击
focus	<a>	onfocus	链接获得输入焦点
	<input>		输入元素获得焦点
	<textarea>		文本域获得输入焦点
	<select>		选择元素获得输入焦点

续表

事　件	标　签	标 签 属 性	说　明
keydown	<body>,表单元素	onkeydown	键按下
keypress	<body>,表单元素	onkeypress	键按下并释放
keyup	<body>,表单元素	onkeyup	键释放
load	<body>	onload	加载完文档
mousedown	绝大部分元素	onmousedown	单击
mousemove	绝大部分元素	onmousemove	用户在元素中移动光标
mouseout	绝大部分元素	onmouseout	光标从所在元素中移出
mouseover	绝大部分元素	onmouseover	光标悬浮于元素之上
mouseup	绝大部分元素	onmouseup	释放左键
reset	<form>	onreset	单击“重置”按钮
select	<input>	onselect	在元素内容中选择文本
	<textarea>		在元素内容中选择文本
submit	<form>	onsubmit	单击“提交”按钮
unload	<body>	onunload	用户退出文档

将事件处理器连接到事件的过程称为注册。传统的注册方式有两种：一是使用 XHTML 标签的属性绑定事件处理器，例如：

```
<input type="button" id="myButton" onclick="myfunc( )" />
```

另一种是使用 JavaScript 对象的属性绑定处理器，例如：

```
document.getElementById("myButton").onclick = function (){ alert('thanks')};
```

程序 4-4 演示了用标签属性绑定事件处理器的方法。

程序 4-4.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 4-4.html:用标签属性绑定事件处理器-->
<head>
  <title>4-4.html!</title>
  <script type="text/javascript" src="nochange.js"></script>
</head>
<body>
<form action="">
  <h3> 中国绿茶订单</h3><table border="1">
  <tr>
    <th>品名</th>
    <th>价格</th>
    <th>订量</th>
    </tr>
  <tr>
```

```
    <th>日照绿茶</th>
    <th>￥300.00</th>
    <td><input type="text" id="rz" size="6" /></input></td>
  </tr>
  <script type="text/javascript">
  </script>
  <tr>
    <th>杭州龙井</th>
    <th>￥900.00</th>
    <td><input type="text" id="hz" size="6" /></input></td>
  </tr>
  <tr>
    <th>信阳毛尖</th>
    <th>￥120.00</th>
    <td><input type="text" id="xy" size="6" /></input></td>
  </tr> </table>
 <p> <!-- 用标签属性绑定事件处理器-->
   <input type="button" value="总价" onclick="computeCost();" />
   <input type="text" size="8" id="cost" onfocus="this.blur();"/></p>
   <p><input type="submit" id="tj" value="提交订单"/>
    <input type="reset" value="重置"/></p>
 </form>
  </script>
</body>
</html>
```

程序运行结果如图 4-5 所示。



图 4-5 程序 4-4 运行结果

下面的程序 4-5 则演示了用对象属性绑定事件处理器的用法。

程序 4-5.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 4-5.html:用对象属性绑定事件处理器-->
<head>
```

```
<title>4-5.html</title>
<script language="javascript">
//处理器
function show(){
var tab=document.getElementById("tab");
var tr=tab.getElementsByTagName("tr");
for(var i=0;i<=tr.length;i++){
if(i%2==0){
tr[i].style.backgroundColor="#ccc";
tr[i].onmouseover=function(){this.style.background="yellow"};
tr[i].onmouseout=function(){this.style.background="#ccc"}
}else{
tr[i].style.backgroundColor="#fff";
tr[i].onmouseover=function(){this.style.background="yellow"};
tr[i].onmouseout=function(){this.style.background="#fff"}
    }
  }
}
<!--用对象属性绑定事件处理器-->
window.onload=show;
</script>
<style type="text/css">
#tab td{width:100px;}
</style></head>
<body><div style="width:500px;align:center;">
<h3>学生信息表</h3>
<table id="tab">
<tr><td>姓名</td><td>学号</td><td>籍贯</td><td>院系</td></tr>
<tr><td>张三</td><td>0891</td><td>山东</td><td>信传</td></tr>
<tr><td>李四</td><td>0892</td><td>四川</td><td>物理</td></tr>
<tr><td>王五</td><td>0897</td><td>云南</td><td>经济</td></tr>
<tr><td>刘琦</td><td>0876</td><td>河北</td><td>管理</td></tr>
<tr><td>高坝</td><td>0856</td><td>广西</td><td>美术</td></tr>
</table></div>
</body>
</html>
```

程序运行结果如图 4-6 所示。



图 4-6　程序 4-5 运行结果

4.8 本章小结

动态网站的开发有 3 个部分，前台、后台和数据库，前台的动态交互就是由 JavaScript 完成的。本章从 JavaScript 的特点开始，依次介绍了 JavaScript 在网页中的嵌入方法、JavaScript 的基本语法、JavaScript 的内置对象、浏览器模型和 JavaScript 的事件处理等内容。JavaScript 是基于对象的语言，自身包含了一系列已经创建完成的对象，程序员只需充分利用这些已经存在的对象，而不需要创建新类。它简单易用，可以直接对用户或客户输入作出响应，无须经过 Web 服务器程序。对用户请求的响应采用事件驱动的方式，无论用户在页面上选择菜单，移动鼠标还是移动窗口，都会引起事件的发生。JavaScript 只依赖于浏览器本身，与操作环境无关，具有良好的跨平台性。学习本章内容，不仅要掌握 JavaScript 的基本语法和内置对象的用法，尤其要掌握利用 JavaScript 在 XHTML 页面上动态写入内容等重要编程技巧。

4.9 练习题

1. JavaScript 怎样嵌入 XHTML 文档？
2. JavaScript 声明变量和函数的关键字分别是什么？
3. Array 对象的 length 属性和该对象中已有元素的实际数目之间是什么关系？
4. String 对象的 indexof 方法和 fontsize 方法的作用分别是什么？
5. 什么是浏览器对象模型？DOM 对象模型是什么？在网上查找两者的关系。
6. 什么是事件？什么是事件源？什么是事件处理器？怎样绑定事件处理器？
7. 怎样利用 getElementById 对 XHTML 元素进行定位？
8. 编程实现选项卡功能。
9. 编程实现图片的放大功能，放大图片随鼠标移动而移动。
10. 编程实现向<div>中写入动态内容的功能。

第5章 服务端语言 PHP

本章重点

- PHP + IIS 的安装配置;
- PHP + Apache 的安装配置;
- PHP 的特点及其在网络中的应用;
- PHP 常用开发工具。

5.1 什么是 PHP

5.1.1 PHP 的概念

PHP 是个人主页：超文本预处理器（Personal HomePage：Hypertext Preprocessor）的缩写。当然这是一个很古老的称呼，已经远远不能反映今天 PHP 的真实能力。PHP 当今已经不仅仅是一个可以用在个人主页上的服务器端脚本语言，而已经成长为一门极为流行、深受 Web 程序员喜爱的、风靡全球的 Web 程序设计语言。它是开源、免费和跨平台的，而且具有高效、简单和安全等特点。Web 开发者能够快速地掌握 PHP 并写出功能强大的服务器端脚本。

5.1.2 PHP 的发展历史

PHP 的创建者是 Rasmus Lerdorf。最初它只是一个用 Perl 语言编写的小程序，名字叫 PHP/FI，用于计算网页访问量。后来 Rasmus 又用 C 语言重新编写，增加了数据库访问功能。Rasmus 免费发布了这个程序的源代码，使得全世界的人都可以免费使用，甚至对其修改、完善。直到今天，PHP 仍然是开源软件领域成功的典范之一。

另外两位对 PHP 有突出贡献的重要人物是 Andi Gutmans 和 Zeev Suraski。1997 年，他们针对 PHP/FI 存在的不足进行了重写，经过 9 个月的测试后，1998 年 6 月，Andi、Rasmus 和 Zeev 联合发布了 PHP 历史上重要的 3.0 版本，这在 PHP 发展过程中有里程碑式的意义。

PHP 3.0 一经推出就大受欢迎，在 PHP 3.0 的顶峰，Internet 上 10%的 Web 服务器上都安装了它。

此后，PHP 快速发展，并在全世界广泛流行起来。PHP 官方又先后发布了 PHP 4、PHP 5 版本，每个版本都有大的改善和提升。尤其 PHP 5.2.0 版本的面向对象功能已经开始成熟，具备了开发大型系统的能力，PHP 在大型电子商务网站已有许多应用。

PHP 在 Web 编程中属于后起之秀，而且也没有大的商业公司作后盾，因此其发展初期并不为国内网站开发人员所重视。最先进入中国的是 PHP 3.0 版本，在相当长的时间内，PHP 在国内使用率很低。但是随着 Internet 在中国的迅猛发展，学习网站开发技术的人越来越多，PHP 以其易学、高效、安全、免费、跨平台等一系列重要优势迅速脱颖而出，吸引了大量的学习者。很多高校已经开设 PHP 课程，各服务器提供商也纷纷提供 PHP 支持。现在浏览国内网站不难发现，PHP 的踪迹无处不在，很多大的网站都是采用 PHP 架构。如今的 PHP 已经与流行的 ASP、JSP、ASP.NET 等并列成为使用广泛的 Web 编程语言之一。

PHP 目前的最新版本是 PHP 6，但它仍处在开发阶段，相比于 PHP 5 基本没有实质性的变化。本教材仍然采用目前比较稳定的 PHP 5 进行介绍。如果想获得更多更新的 PHP 相关资料，可以到 PHP 官方网站 http://www.php.net 上浏览。

5.2 PHP 可以做什么

PHP 脚本主要用于以下 3 个领域。

1. 服务端脚本

这是 PHP 最传统也是最主要的目标领域。所谓服务器端脚本是指运行在服务器端的网页程序。Web 服务器每次接收到访问请求后，执行（或解释）此程序，将程序的运行结果发送到客户端浏览器，而不是传统的 XHTML 静态网页中的不经处理直接发送。这种方法可以突破 XHTML 标记性语言的局限性，为网页加入可以实现功能动态化的程序。网页中数据来源于普通文件或数据库。要使用 PHP 进行服务器端脚本开发，只需要将自己的计算机配置成为一台支持 PHP 的 Web 服务器即可。后面的章节中详细地介绍 PHP 开发环境配置的方法。

2. 命令行脚本

可以编写一段 PHP 脚本，并且不需要任何服务器或浏览器来运行它。通过这种方式，只需要 PHP 解析器来执行。这种用法对于依赖 cron（UNIX 或 Linux 环境）或 Task Scheduler（Windows 环境）的日常运行的脚本来说是理想的选择。这些脚本也可以用来处理简单的文本。

3. 桌面应用程序

对于有着图形界面的桌面应用程序来说，PHP 或许不是一种最好的语言，但是如果用户精通 PHP，并且希望在客户端应用程序中使用 PHP 的一些高级特性，可以利用 PHP-GTK 编写这些程序，还可以采用这种方法编写跨平台的应用程序。

目前 PHP 主要应用于 B/S 架构的网络应用程序开发任务。从一般的网站新闻程序、留言本、用户注册与登录、投票调查、计数器、网上登记、网上查询到大型论坛程序、大型网上电子商务平台、网上办公系统、信息管理系统（IMS/CMS）等。目前全球 5000 万互

联网网站中，有 60%以上使用着 PHP 技术； 2011 年 PHP 从业人数将增加 42%，远超 Java 的 13%和.NET 的 24%；PHP 也入选是全球五大最受欢迎的编程语言，并且是唯一入选的脚本语言；国内 80%以上的动态网站都在使用 PHP 开发；AlexaTOP500 中国网站排名，有 394 家使用了 PHP 技术，比例为 78.8%。全球知名互联网公司 Google、Yahoo!和国内知名网站新浪、百度、阿里巴巴、腾讯、网易等均采用了 PHP 技术。

最近几年，各类新的网络技术的兴起也大大丰富了 PHP 的能力。带动了 PHP 周边技术的发展，从而进一步开辟了 PHP 的应用领域，如模板技术（Template）、网页异步通信（Ajax）等在 PHP 中都得到了应用。

5.3　PHP 有哪些特性

5.3.1　PHP 的特点

PHP 作为一门 Web 开发语言，具有区别于其他语言的诸多特点。本节重点探讨 PHP 语言的特点。PHP 自产生以来一直都在发展中应用，在应用中发展，因为 PHP 不仅有着其他同类脚本所共有的功能，而且更有它自身的特点。

PHP 的主要特点如下：

（1）完全免费。使用 PHP 进行 Web 开发无须支付任何的费用。

（2）代码完全开放。PHP 是开放源代码的，这就意味着所有的 PHP 程序代码都可以免费地使用和交流。

（3）语法结构简单。PHP 大量结合了 C 语言的特色，同时又集成了当前流行的面向对象的编程理念，坚持以基础语言开发程序，编写方便易懂。对于以前接触过 C 语言的用户来说，只需要了解 PHP 的基本语法，再掌握些 PHP 独有的函数，就可以轻松踏上 PHP 程序设计之旅。比同类语言如 JSP（Java）、ASP.NET（C#），PHP 具有明显的简易优势。

（4）功能强大。PHP 的强大功能在前面的章节中已经介绍过，这里不在赘述。

（5）强大的数据库支持。PHP 几乎支持所有的主流数据库，如常用的 MySQL、SQL Server、Oracle 等。

（6）代码执行效率高。与其他 CGI 比较，PHP 消耗更少的系统资源，尤其当 PHP 作为 Apache 服务器的内嵌模块运行时，服务器除了承担脚本解释负荷外，无须承担其他额外操作。

（7）安全性高。作为 Web 开发语言，安全性如何是一项不可或缺的重要指标。因为 PHP 本身是开源的，这就使得全世界的人都可以对代码进行研究，进而尽可能多的发现存在的问题和错误，并及时修正。PHP 是公认的具备高安全性的语言。迄今 PHP 尚未发现可以造成重大破坏的安全漏洞。

5.3.2　PHP 与其他 CGI 的比较

很多网站开发的初学者都会问同一个问题：Web 开发中哪种语言最好？应该去学习哪一门语言？其实这个问题很难回答。每种语言都有其自身的特点和优势，相比之下也各有

劣势。下面把目前较为流行的几种 Web 开发语言（技术）的各个指标进行一个类比，如表 5-1 所示，使读者有一个直观比较和认识。

表 5-1　几种 Web 开发语言（技术）的比较

	PHP	ASP	CGI	JSP	ASP.NET	ISAPI
操作系统	均可	Win32	均可	均可	Win32	Win32
Web 服务器	多种	IIS	均可	数种	IIS	IIS
执行效率	快	快	慢	很快	很快	很快
稳定性	佳	中等	高	佳	佳	差
开发时间	短	短	中等	较长	中等	长
程序语言	PHP	VB	多种	Java	C#	C/Delphi
网页结合	佳	佳	差	差	中	差
学习门槛	低	低	高	高	高	高
函数支持	多	少	不定	多	多	少
系统安全	佳	差	佳	佳	佳	尚可

必须说明的是，表 1-1 中的数据并不是笔者通过实测总结出来的，实际上要对一种语言的各种性能进行很具体的实测很复杂，而且相当困难。尽管没有经过实测，但是就笔者的使用经历，结合其他资料中的观点，以及目前业内比较广泛的共识，仍然可以给出一个大体的评估。

5.4　PHP 常用开发工具

编写 PHP 代码的工具很多，常用的网页编辑器有 FrontPage、Dreamweaver，常用的文本编辑器有 UltraEdit，甚至用 Windows 自带的记事本都可以进行书写源代码。当然，还有专门的 PHP 开发工具，如 PHPEdit、PHPCoder、Dev-PHP 等。Zend 公司也推出了一个功能强大的集成化安装环境 Zend Studio。借助这些开发工具可以使用户的开发事半功倍。下面简单介绍各种工发工具。

1. EditPlus

EditPlus 速度快，支持多种语言的语法加亮，是个简易的编辑器。

2. Dev-PHP

用 Delphi 开发的 Opensource 的 PHP 开发工具，较好地集成了 PHP 解析器和 PHP-GTK 库，性能和稳定性都很好；只是在团队合作上比较薄弱，没有 SCC 和 project 的功能。完全让用户有理由舍弃掉 EditPlus。

3. PHPCoder

一个优秀的 PHP 开发工具，支持语法加亮、函数提示，调试功能丰富，有项目管理功能，而且此工具体积小，还有绿色免安装版本，运行速度极快。

4. Microsoft FrontPage

FrontPage 是微软公司推出的 Office 系列中的一款制作网页的软件，简单易学，容

易上手，有 Word 操作经验的人学起来会觉得很容易。FrontPage 目前最新版本为 2007，使用较广的版本是 2003。对于以前用 FrontPage 开发网页过程中出现的垃圾代码，相信用过的读者肯定深有体会。FrontPage 2003 以后这一问题得到了很大的改善。比以前的版本，FrontPage 2003 的功能更强大，界面更友好，产生的垃圾代码更少，开发效率更高。

5. Adobe Dreamweaver

作为名扬天下网页三剑客之一的 Dreamweaver，在网站的设计与开发上功能强大，是一款大型的、普遍认为功能在 FrontPage 之上的网页集成开发环境。Dreamweaver 支持 PHP 代码高亮显示，有一定函数提示功能。Dreamweaver 原为 Micromedia 公司的产品，后被 Adobe 公司收购。目前最新版本为 Dreamweaver CS5，其运行界面如图 5-1 所示。

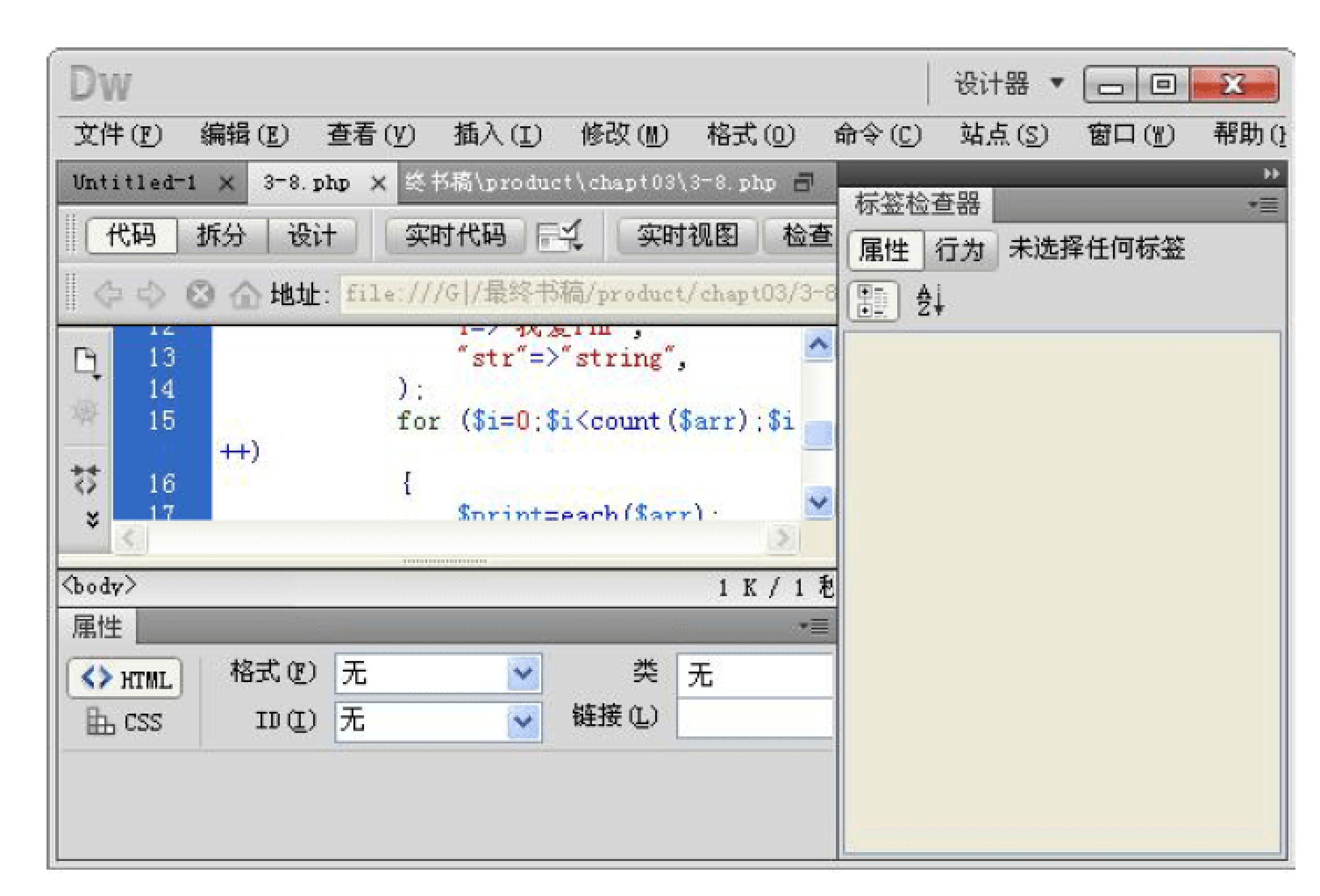


图 5-1　Adobe Dreamweaver CS5 运行界面

6. Zend Studio

Zend Studio 是 Zend 公司推出的 PHP 专业级开发环境。其功能强大，界面友好。它集成了代码编辑、调试和加密多项功能。代码编辑部分拥有项目管理、代码高亮显示、函数提示、代码自动完成、代码整理、断点调试等一系列功能，还集成了多款数据库的可视化管理功能，是目前专业用于 PHP 开发的最好的 IDE 环境。

7. UltraEdit

UltraEdit 是一款功能强大的文本编辑器，可以编辑文字、Hex、ASCII 码，可以取代记事本，可同时编辑多个文件，而且即使开启很大的文件速度也不会慢。最新版本的软件修正了老版本存在的一些 Bug，并新增了二十余项新功能。界面如图 5-2 所示。

8. NotePad（记事本）

这是 Windows 自带的记事本，如果只需要对源代码做很少的修改时可以用它。它占用内存极少，运行速度快，功能较简单。当然，随着微机硬件配备的不断提高和其他专用软件的兴起，它已慢慢淡出了视线。在使用记事本编写好程序存盘的时候应注意，在文件名一栏手动输入文件名字，后缀为 php，保存类型应选择“所有文件”如图 5-3 所示；否则会自动加上后缀名 txt，保存为文本文件，无法正常运行。

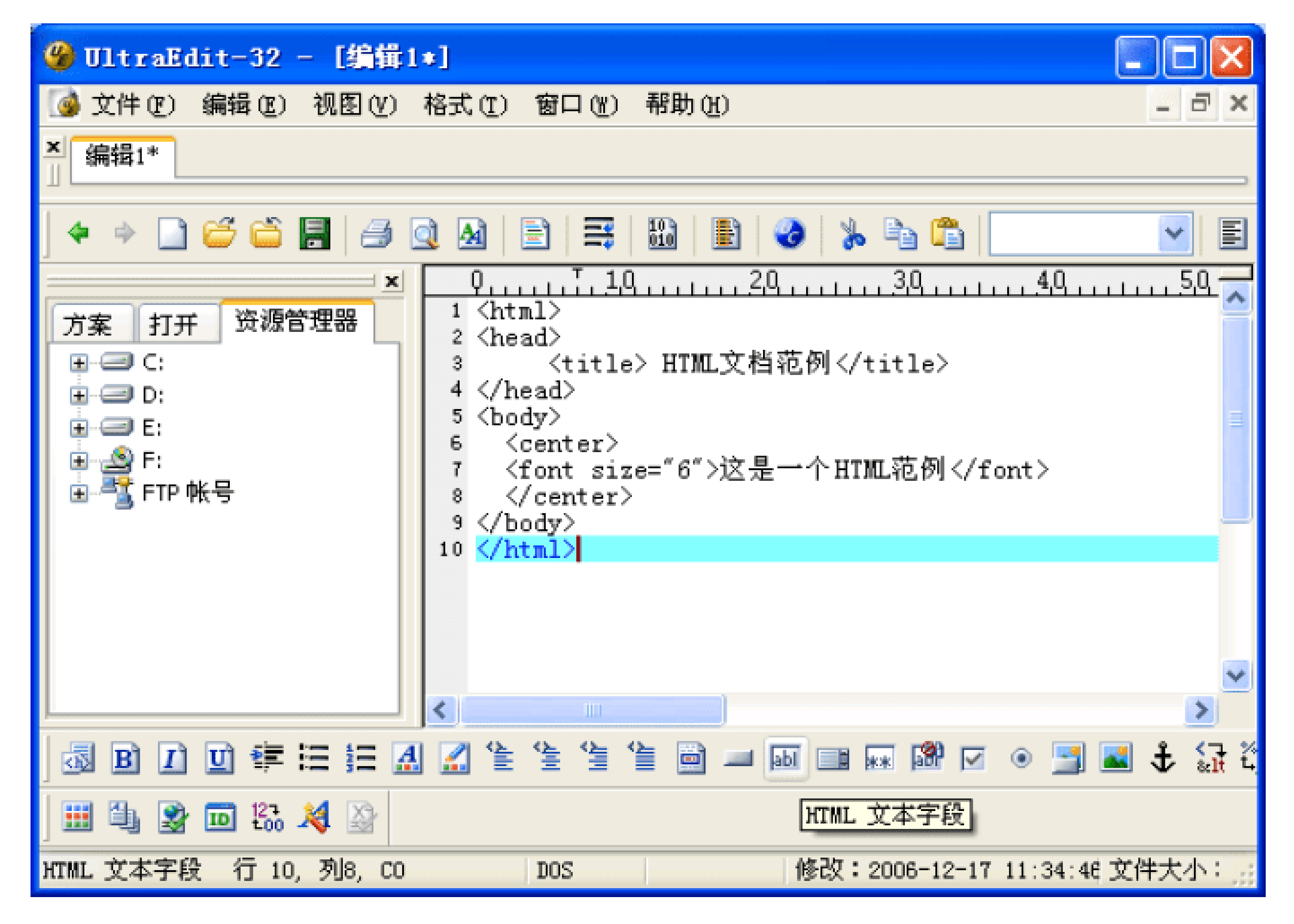


图 5-2　UltraEdit 工具界面

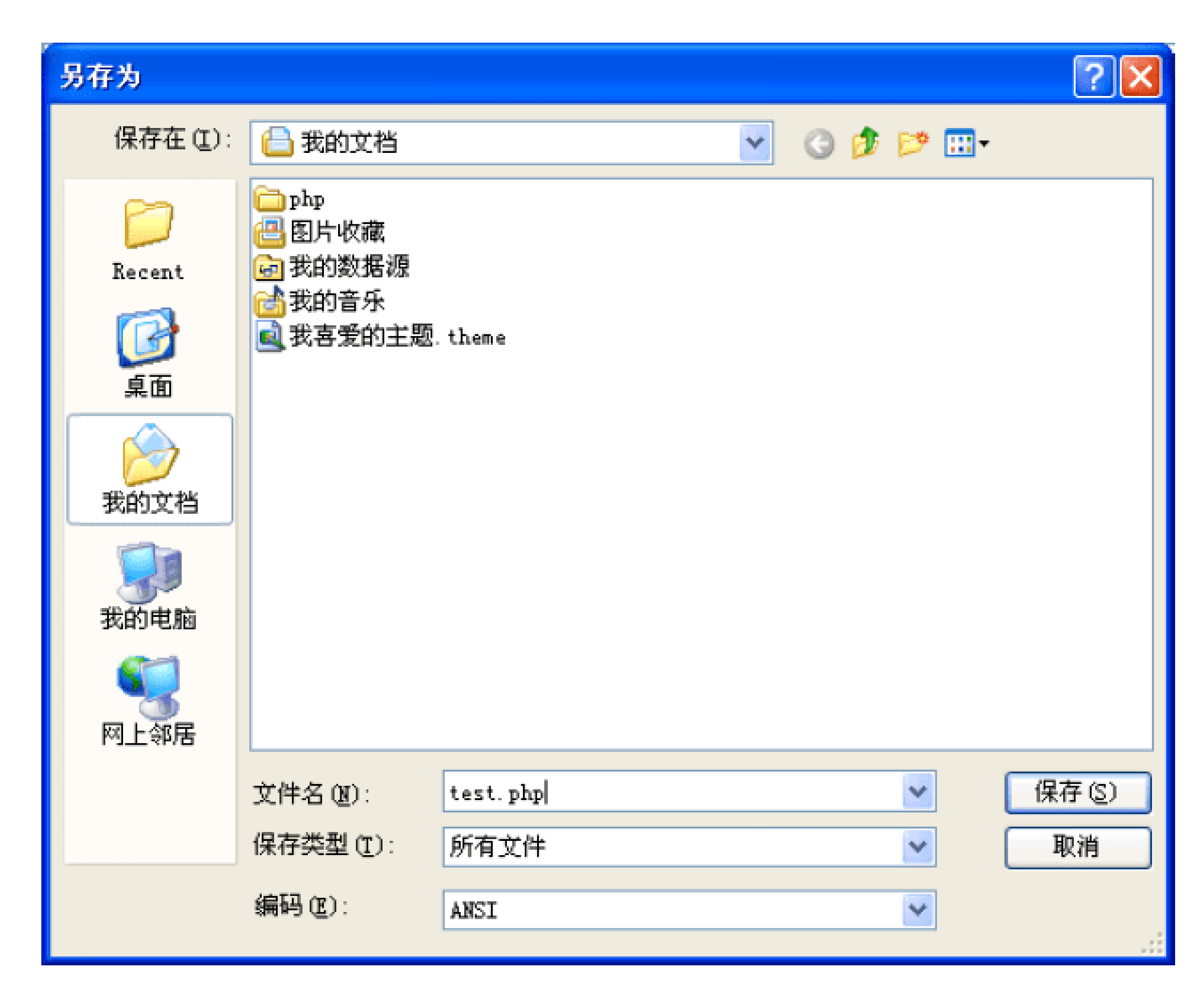


图 5-3　用记事本编写 PHP 代码并存储

当然，能开发 PHP 的工具绝非只有前面提到的几种，这里只是从每种类别的工具中选出具有代表性的来介绍。这么多的工具，肯定有一款比较适合自己，就笔者而言，可视化所见即所得的开发界面还是推荐 FrontPage 或 Dreamweaver。专业 PHP 开发工具推荐使用 Dev-PHP 或 Zend Devlopment Environment。对于本书的读者，如果是 PHP 的初学者，刚刚踏入 PHP 编程的大门，建议暂时不要使用任何编辑工具，否则很容易被这些工具复杂的操作界面吓倒。对于本书中的绝大多数例子来说，代码量都比较小，完全可以直接采用 Windows 的记事本进行编写、调试。虽然这可能带来一些弊端，如缺乏语法与拼写检查、没有函数提示等，但对于初学者来说，从最基础的学起，在代码调试过程中积累经验，对于最终养成良好的编程习惯，具有积极的作用。

5.5　PHP 程序运行原理

XHTML 网页的基本运行原理是，客户端通过浏览器向服务器发出页面请求，服务器收到请求后直接将所请求的页面发回给客户端，然后客户端就能在浏览器中看到页面的显示效果。这是一个比较简单、直接的过程，只需要一台安装了 Web 服务软件的服务器就能完成。

PHP 和其他服务器端嵌入式脚本语言一样，需要首先搭建专门的服务器环境。只有配置好服务器环境，一台服务器才能运行 PHP 网站。PHP 网站和用其他语言开发的动态网站运行原理基本相同，其简要流程如图 5-4 所示。

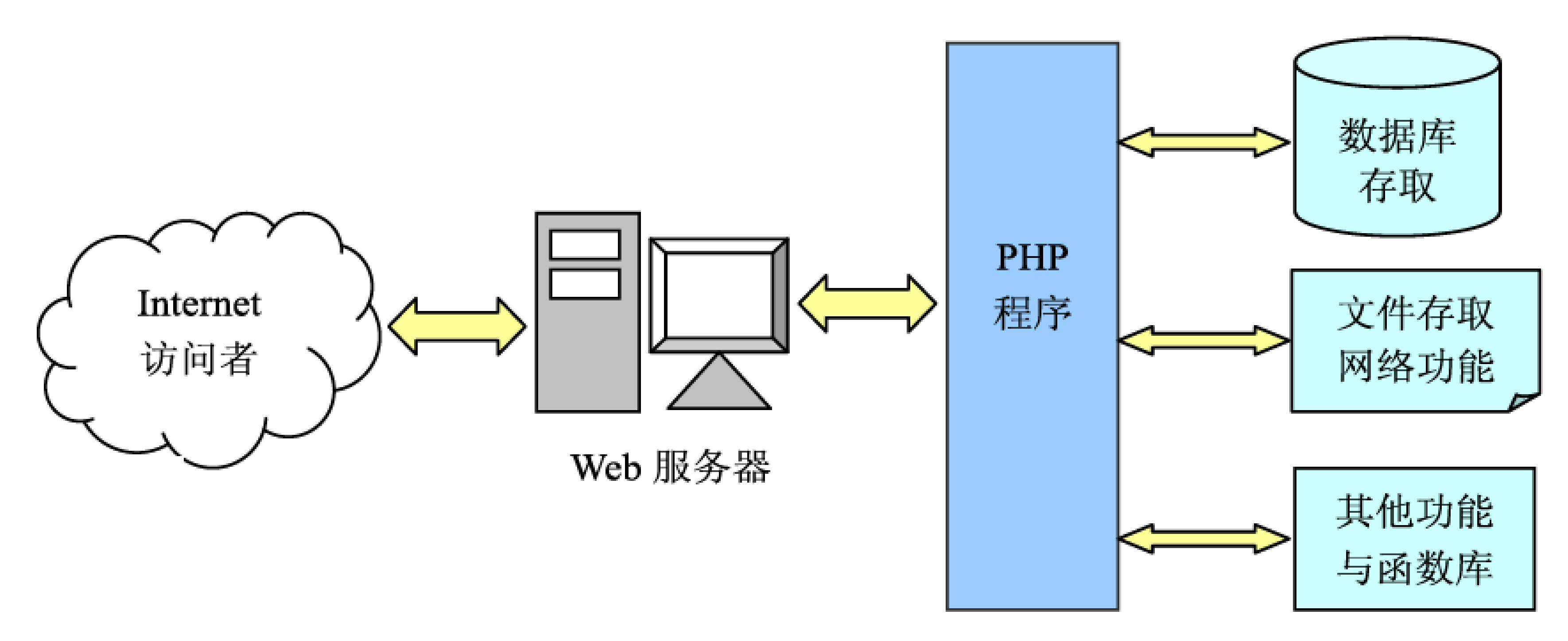


图 5-4　PHP 的运行流程图

可以看出，PHP 程序通过 Web 服务器接收访问请求，在服务器端处理请求然后再通过 Web 服务器向客户端发送处理结果。在客户端接收到的只是程序输出的处理结果，是一些 XHTML 标记，而无法直接看到 PHP 代码。这样，能够很好地保证代码的保密性和程序的安全性。此外，在服务器端运行代码还可以降低对客户端的要求，客户端不需要配置 PHP 环境，只需要安装普通浏览器即可浏览 PHP 网站。

PHP 是对服务器功能的扩展，服务器上安装了 PHP，就可以用它来实现很多应用。接下来开始介绍 PHP 服务器环境配置的方法。

5.6　PHP 安装前的准备

5.6.1　软件和硬件环境

安装 PHP 和安装其他软件一样，都需要软件和硬件环境。对于 PHP 来说，硬件的要求非常简单，在学习阶段只要有一台普通计算机就足够了。软件方面则需要根据自己的情况进行选择，主要从操作系统、Web 服务软件两个方面考虑。

PHP 能够运行在目前所有的主流操作系统上，包括 Linux、UNIX 及其各种变种（包括 HP-UX、Solaris 和 OpenBSD）、Microsoft Windows 系列、Mac OS X、RISC OS 等。PHP 在

这些平台上的安装步骤大同小异，本书主要以 Windows 平台为例介绍 PHP 的安装和使用，同时简要介绍在 Linux 系统上的安装。PHP 具有跨平台的特性，因此在 PHP 开发阶段使用什么样的操作系统并不重要，因为开发出来的程序可以很容易地移植到其他操作系统中。

除了操作系统外，和 PHP 的安装息息相关的就是 Web 服务软件，也称 Web 服务器。PHP 可以支持大多数的 Web 服务器，包括 Apache、Microsoft Internet Information Server（IIS）、Personal Web Server（PWS）、Netscape、iPlant server、Oreilly Website Pro Server、Caudium、Xitami、OmniHTTPd 等。这些 Web 服务器各有特点，目前以 Apache 和 IIS 的使用最为广泛。本章将分别介绍 PHP 在 Apache 和 IIS 上的安装方法。

综上所述，要安装 PHP，首先只有保证计算机的操作系统和 Web 服务器已经安装并能够正常工作，才可以开始 PHP 的安装。

5.6.2 获取 PHP 安装资源包

要安装 PHP，当然首先要获取 PHP 的安装资源包。这个资源包中包括了安装和配置 PHP 服务器的一切文件以及大量 PHP 扩展函数库。PHP 安装资源包的获取有很多途径，如登录 PHP 官方网站下载，或通过其他网站下载。PHP 是免费软件，网上大量的软件下载站点都收录了此软件，可以通过搜索引擎搜索该软件，然后直接下载。

值得注意的是，很多网站提供的 PHP 安装资源包并不规范和全面。有些 PHP 安装软件是由第三方开发的，并没有经过 PHP 官方的认可和授权。有的安装包为了缩小体积还删掉了大部分扩展函数库，给以后的使用带来不便。因此，建议直接去 PHP 官方网站下载 PHP 安装资源包，并用手工方法安装和配置 PHP 服务器。只有这样才能保证所使用的 PHP 的完整、全面和权威。

登录 PHP 官方网站 www.php.net，如图 5-5 所示。



图 5-5 PHP 官方网站

单击 Downloads 链接打开 PHP 的下载页面，该页面列出了最新的 PHP 安装资源包，如图 5-6 所示。单击 Windows Binaries 中的 PHP 5.1.6 zip package 项。

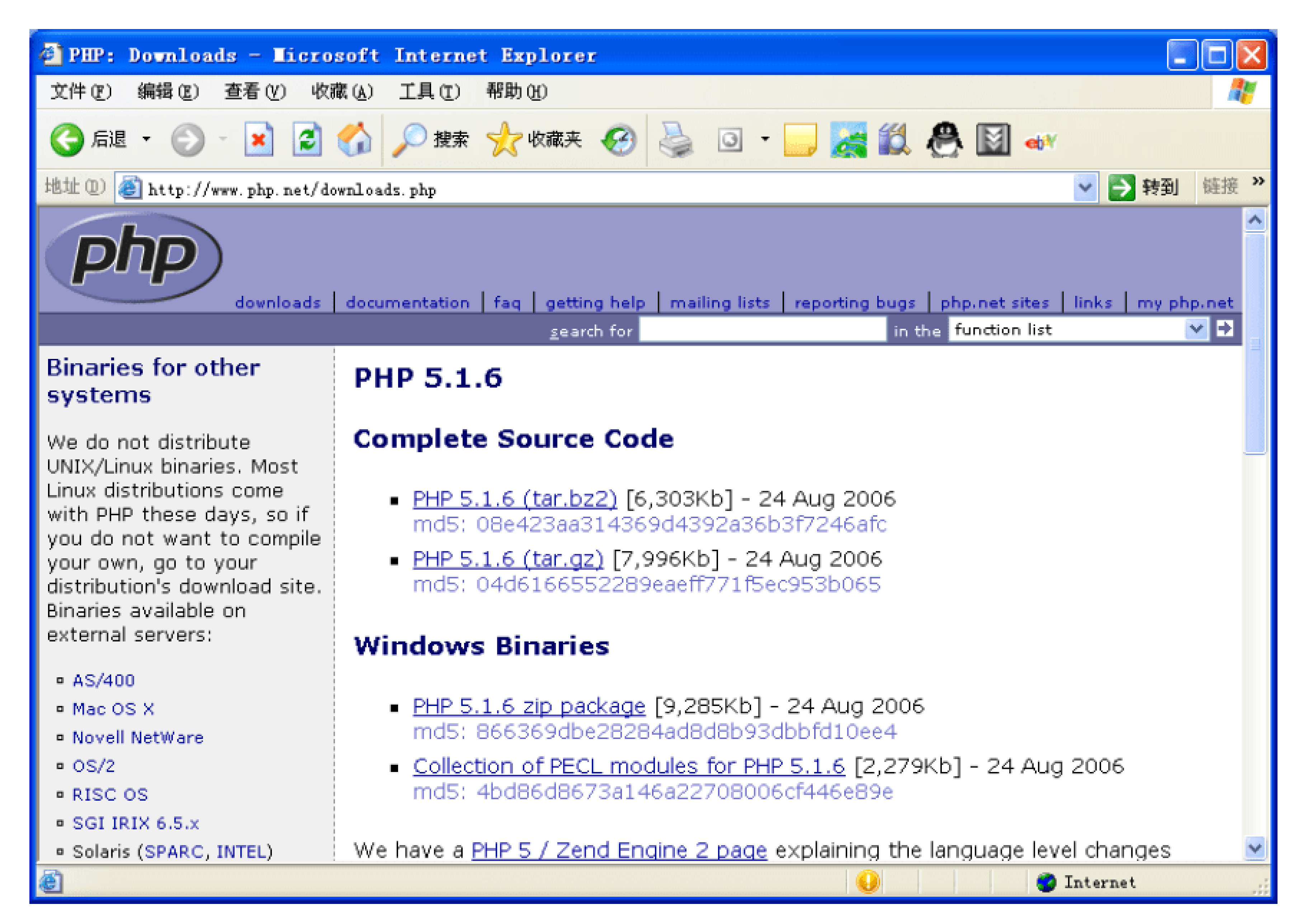


图 5-6　PHP 官方下载页面

此时出现一个镜像站点选择页面，一般网页会自动判断用户的地域并给出推荐的镜像站点，直接根据推荐站点下载即可。如果推荐站点无法下载，可以选择一个离自己较近的镜像站点下载，如图 5-7 所示。

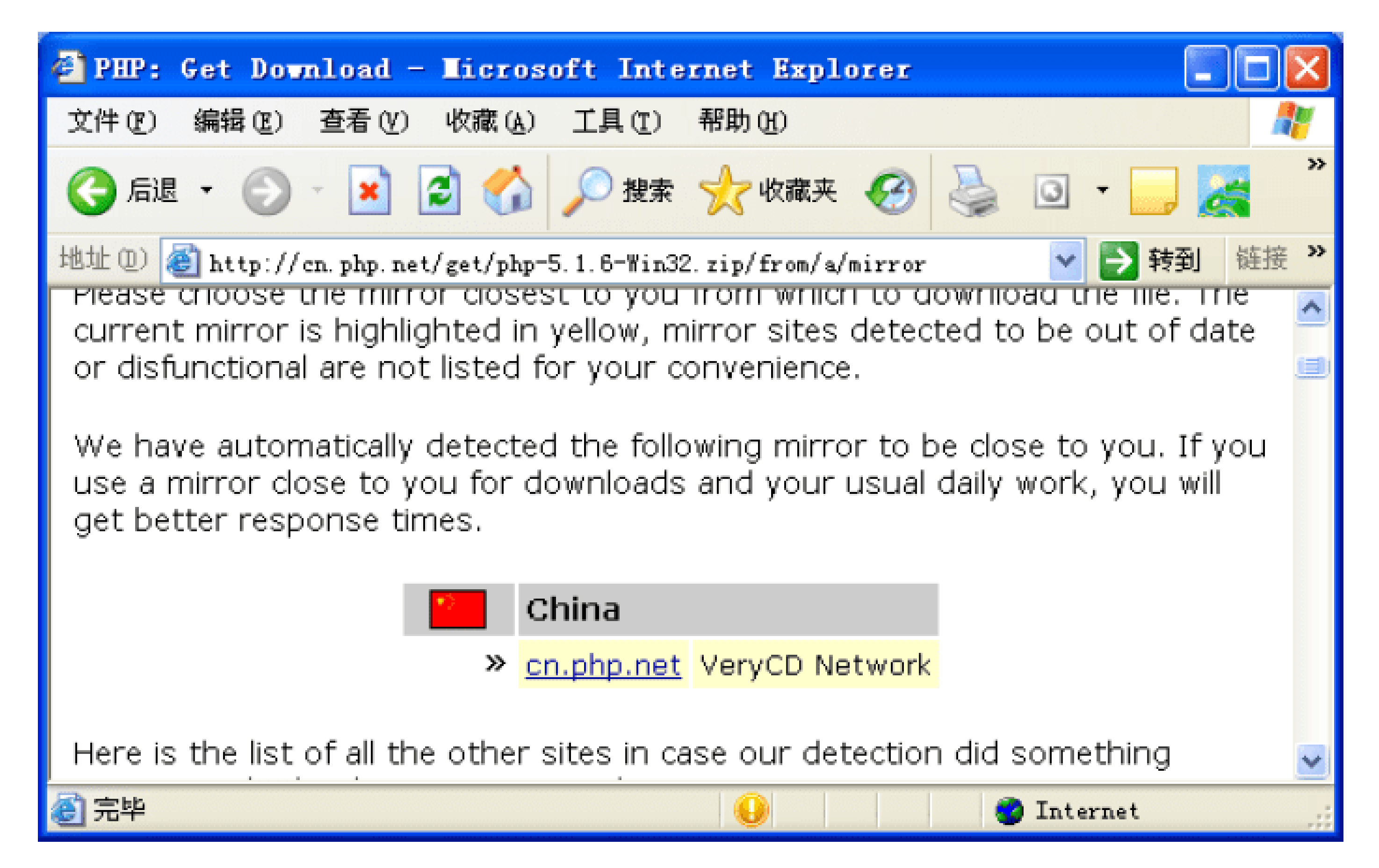


图 5-7　下载 PHP

下载后，得到的是一个名为 php-5.1.6-Win32.zip 的压缩包，这就是 PHP 安装资源包。通过此压缩包的名字就能看出其版本号为 5.1.6，其适用平台为 Windows 系列。如果下载到的资源包与此名字不符，应考虑是否单击了错误的链接。

到此为止，已经做好了 PHP 服务器搭建的准备工作，下面可以开始安装了。

5.7 Windows 下 PHP 的安装与配置

5.7.1 Windows XP 下安装 PHP

虽然 PHP 正常情况下运行于服务器操作系统上，而 Windows XP 不能算是服务器操作系统，但是鉴于目前 Windows XP 操作系统使用的广泛性，很多开发者乐于在 Windows XP 下做 PHP 开发，因此首先介绍 Windows XP 下 PHP 的安装配置。

1. 安装 IIS

在 Windows XP 环境下，可以采用 Windows XP 自带的服务器组件 IIS（Internet Information Server）作为 Web 服务软件。IIS 是目前使用较为广泛的 Web 服务器之一，操作简单，使用方便，功能强大。IIS 由微软公司开发，目前也只能运行在微软公司的 Windows 系列操作系统上，包括 Windows 2000、Windows XP 及 Windows Server 2003 等版本。值得注意的是，Windows XP Home 版下无法直接安装 IIS，因此如果要在 Windows XP 下开发 PHP，建议不要使用 Home 版。

部分版本的 Windows 操作系统，如 Windows 2000 Server、Windows Server 2003 等，安装系统时默认自动安装 IIS，其他版本的则默认不安装。要检查自己的操作系统是否已经安装了 IIS，可以打开控制面板，找到“管理工具”并打开，看里面是否有“Internet 信息服务管理器”的快捷方式。如果有，则 IIS 已经安装；否则可能没有安装。另外一个快捷的方法是直接选择“开始”→“运行”命令，打开“运行”对话框，输入 inetmgr 命令，按 Enter 键，看是否能打开 IIS 管理界面，如果能打开，则表示已经安装了 IIS，否则说明没有安装。

如果系统还没有安装 IIS，首先需要安装。安装的方法很简单，打开控制面板后，打开“添加或删除程序”，然后单击左侧所列选项中的“添加/删除 Windows 组件”，稍后会出现“Windows 组件向导”对话框，如图 5-8 所示。



图 5-8 “Windows 组件向导”对话框

勾选“Internet 信息服务（IIS）”复选框，单击“下一步”按钮，即可开始安装 IIS。在此过程中可能会提示插入系统安装光盘。等待安装过程结束，IIS 便安装成功。这时可以通过上面介绍过的方法验证一下是否已经安装成功。

2. 安装 PHP

在确认自己的系统已经安装 IIS 之后，就可以开始 PHP 的安装了。实际上本安装方法同时适用于 Windows 2000。

1）解压和复制

在此之前已经下载了 PHP 安装资源包 PHP 5.1.6-win32.zip，现在要做的第一件事就是将这个资源包解压缩，解压缩后会得到一个文件夹，文件夹中放着运行 PHP 所需要的一些文件和文件夹。接下来将这些文件和文件夹复制到 PHP 的安装目录。PHP 的安装目录可以根据自己的情况随意设置，这里设为 C:\PHP 5\，如图 5-9 所示。

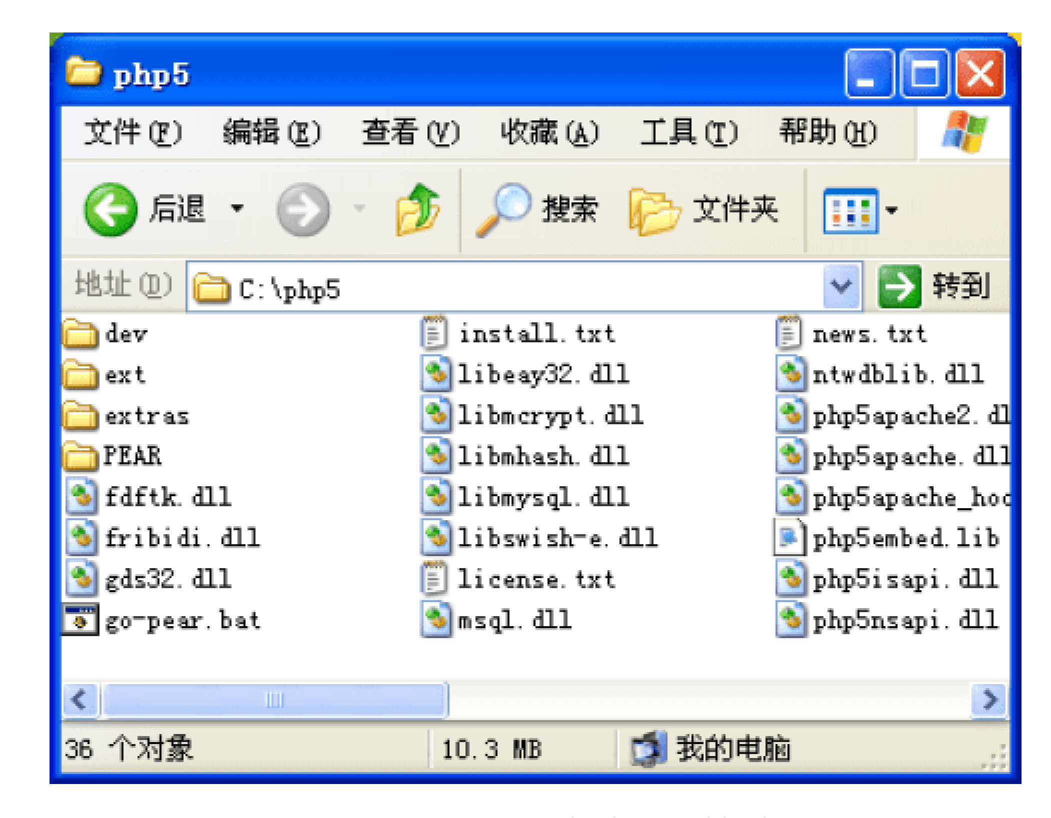


图 5-9　PHP 安装文件夹

2）配置 php.ini

在安装目录下，找到一个名为 php.ini-dist 的文件。这个文件就是 PHP 的配置文件，里面存储了所有 PHP 运行时所需要的参数。通过修改这个文件可以实现对 PHP 的配置了。注意，这个文件的扩展名是 ini-dist，需要首先把扩展名修改为 ini，然后打开记事本。打开之后，可以看到许多令人眼花缭乱的参数，但是需要用户修改的参数很少。php.ini 中的参数形式为：

参数名=参数值

如 session.auto_start=0，前面的 session.auto_start 就是参数名，0 就是参数值。配置 PHP 参数就是找到相关参数名，然后按照需要修改其值。

（1）找到 extension_dir 参数，此参数设置 PHP 扩展函数库的查找路径。将其值修改为“安装目录\ext\”。本例中为 C:\PHP 5\ext\。

（2）找到 session.save_path 参数，这个参数用来设置 Session 的保存路径，在后面的章节中会详细讲解 Session 的使用。可以指定任意一个目录存放 Session。在本例中，设置为 C:\PHP 5\sessions\。当然首先必须创建这个目录，接着在下面几行找到 session.auto_start 参数，对于初学者，建议将它设置为 1，这样服务器一启动，便自动支持 Session，在编写代码时就可以不必手工开启它。

（3）找到 file_uploads 参数，还有下面连续的两个 upload_tmp_dir 和 upload_max_filesize，这 3 个参数是用来控制有关文件上传的。表示的意思分别是，是否允许文件上传（on/off）、上传文件的暂存路径、上传文件的最大字节数。可以根据自己的需要来修改和填写，也可以使用默认值。最好填写 upload_tmp_dir，保证以后可以用 PHP 实现上传文件功能。可以在 C:\PHP 5\下创建一个 UPLOADS 文件夹，并将 upload_tmp_dir 设置为“C:\PHP 5\UPLOADS\”。此处的文件夹可以任意指定，但必须保证指定的文件夹存在，如果采用了 NTFS 文件系统，还要保证此目录有足够的访问权限。

（4）找到“;php_MySQL.dll”和“;php_gd2.dll”，将前面的分号去掉。这样可以保证能够正常使用 MySQL 数据库函数和图形函数。

至止，PHP 的配置已经足够初学者使用很长时间了。当然 php.ini 中还有很多重要的配置，这里没有一一提到。在没有深入理解这些配置的用途之前，还是采用默认值比较好。如果读者对此感兴趣，可以参考 PHP 手册以及 PHP 安装说明文件来了解更多的参数。值得注意的是，对这个文件的修改必须慎重，因为这个文件直接关系到 PHP 的运行。

3）复制文件

将修改好的 php.ini 复制到系统目录下。在 Windows XP 下，系统目录为 C:\WINDOWS\（视操作系统安装位置而定）。在 Windows 2000 下为 C:\WINNT\。

将 PHP5 目录下的 PHP 5ts.dll、libMySQL.dll 两个文件复制到系统目录下的 System32 子目录中。如 C:\WINDOWS\SYSTEM32\。

复制这些文件的主要目的是使系统和 IIS 在启动时就能够载入这些文件，保证一开机就能使用。可以将 PHP 目录添加到系统环境变量中也能达到同样的目的。这种方法不再详述，读者如有兴趣可以查阅相关资料。

至此，PHP 的配置告一段落，不过目前还无法编写程序测试安装的效果，还需要将 PHP 与 IIS 建立关联，至此服务器才算搭建完毕。需要告诉 IIS，遇到 php 类型的文件请求时如何处理。

3. 将 PHP 与 IIS 建立关联

只有 PHP 和 IIS 建立了关联，才能在 IIS 接收到对 PHP 的页面请求时将处理权转交给 PHP。Windows XP 的 IIS 版本为 5.1，以此为例来说明将 PHP 和 IIS 建立关联的方法。

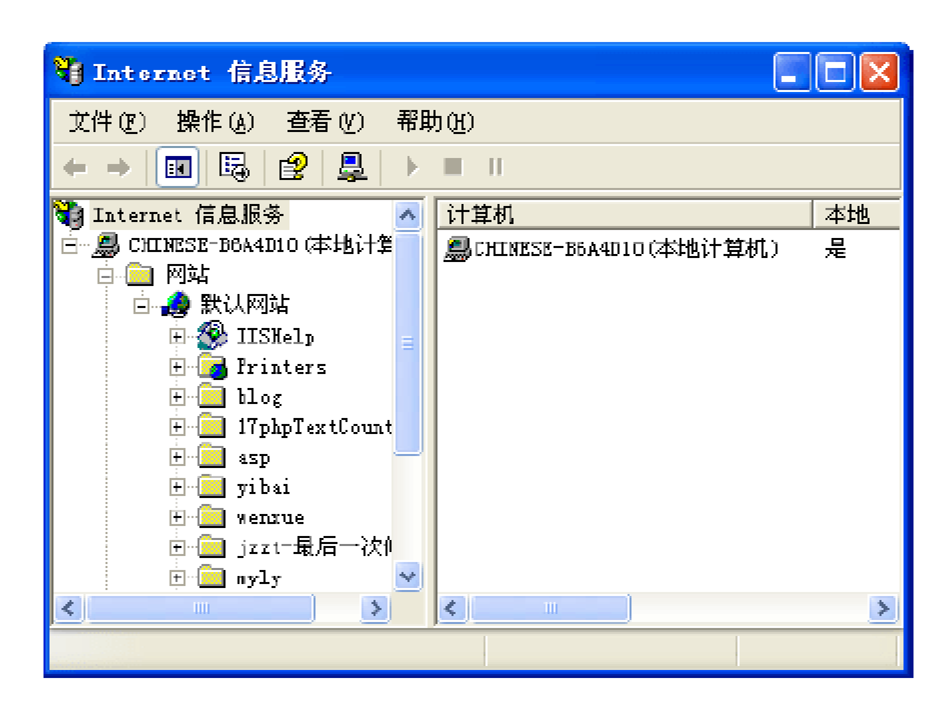


图 5-10　IIS 管理器

打开控制面板，找到“管理工具”→“Internet 信息服务”，或直接选择“开始”→“运行”命令，在“运行”对话框中输入 inetmgr 命令，按 Enter 键，打开 IIS 管理器，如图 5-10 所示。

在管理器左侧的目录树中找到“默认网站”，右击，在弹出的菜单中选择“属性”命令，打开“默认网站 属性”对话框。单击对话框上方选项卡中的“主目录”标签，如图 5-11 所示。

单击“配置”按钮，进入“应用程序配置”对话框，单击“映射”标签，如图 5-12 所示。

单击“添加”按钮，添加一个.php 文件扩展名的映射。在打开的“添加/编辑应用程序扩展名映射”对话框中，单击“浏览”按钮，找到刚才安装 PHP 的路径 C:\php 5\php 5isapi.dll（如果打开了这个目录，没有找到这个文件，那一定是在打开文件对话框中没有选择文件类型为“所有文件”或*.dll 文件），然后在下面的“扩展名”中填入想使用的 PHP 程序的扩展名，可以使用 PHP，PHP 5 等。为了尽可能提高可移植性，推荐使用 PHP。另外，还要注意要勾选“脚本引擎”和“检查文件是否存在”复选框，如图 5-13 所示。

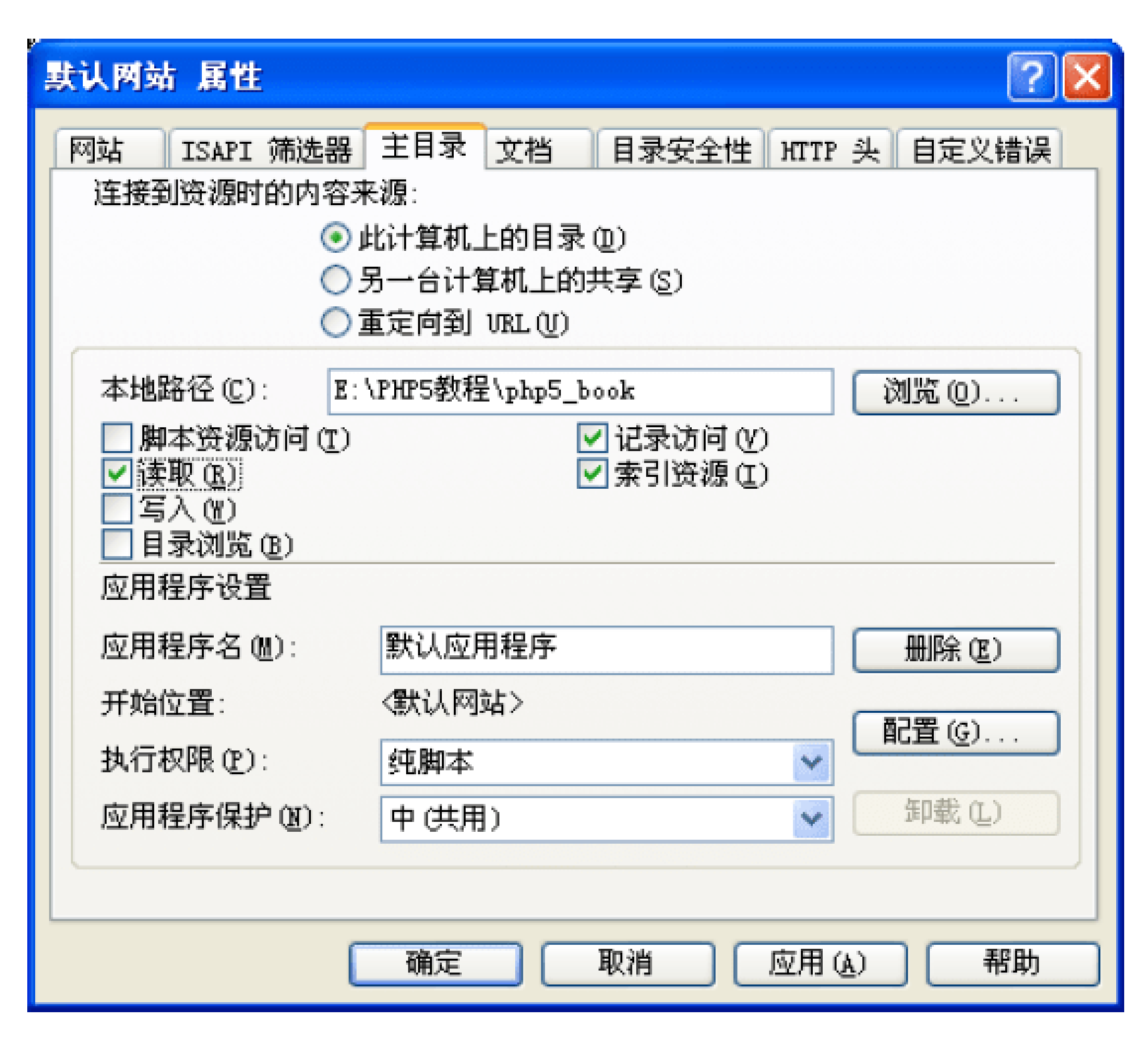


图 5-11　“默认网站 属性”对话框

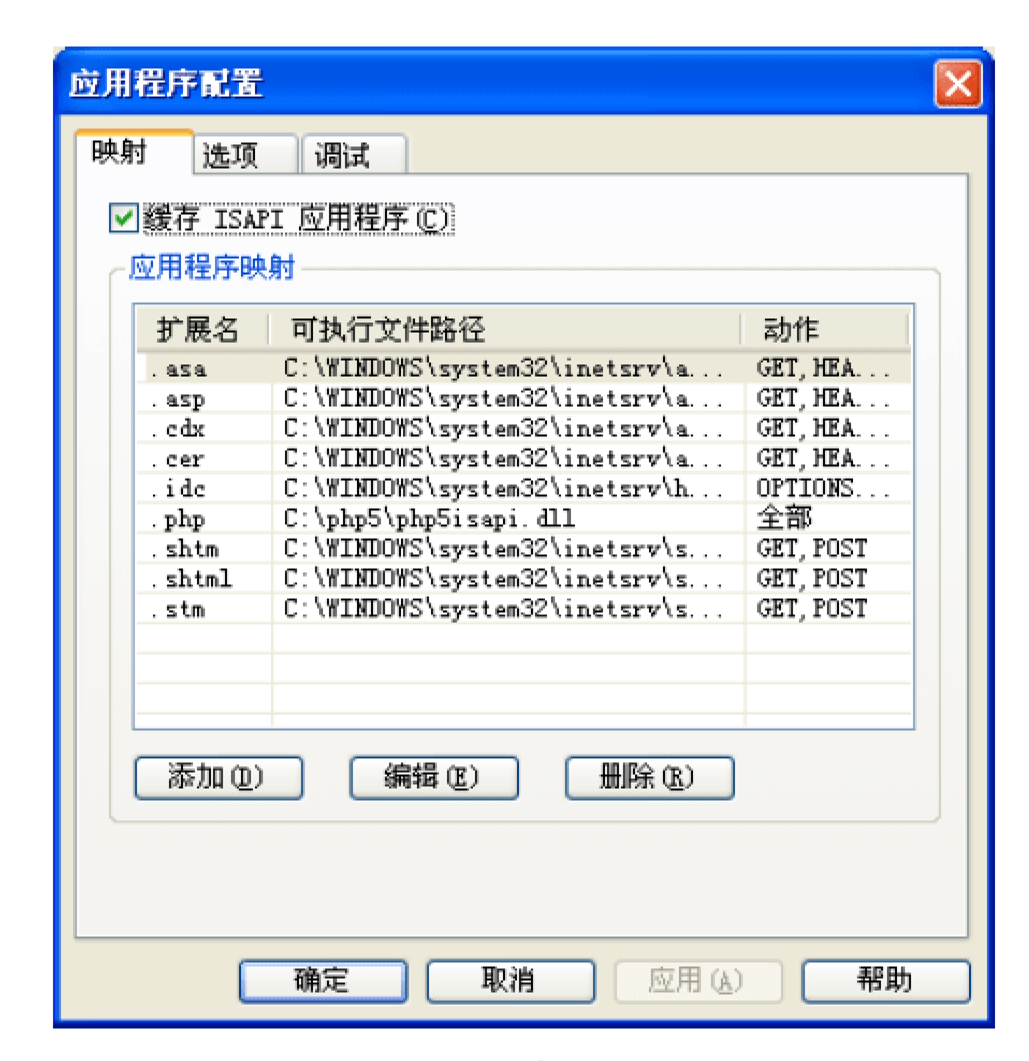


图 5-12　应用程序“映射”配置



图 5-13　“添加/编辑应用程序扩展名映射”对话框

依次单击“确定”按钮关闭所有对话框，关闭 IIS 管理器，至此在 Windows XP 下的 PHP 配置已经完成了。现在需要完成最后一步操作——重新启动 IIS（如果觉得重新启动 IIS 比较麻烦，就可以直接重新启动计算机）。在“运行”对话框中输入 iisreset 命令，按 Enter 键。稍后 IIS 就已重新启动。至此，IIS 与 PHP 的关联工作完成。

4. 运行测试

现在，完全可以编写第一个 PHP 程序了。使用任意文本编辑软件（如 Windows 的记事本、UltraEdit 等）新建一个文本文件，输入下列代码：

```
<?php
phpinfo();
?>
```

将此文件保存为 show_info.php，注意此文件一定要保存为 php 文件。初学者很容易犯的一个错误就是将 PHP 文件存为了 txt 文件，或者.php.txt 文件。这些文件都无法运行，一定要保证文件扩展名为 php。

将此文件保存到 IIS 的主目录下。IIS 默认主目录为 c:\intepub\wwwroot\。可以在 IIS 中自行设置这一目录，将其指到其他位置。

将此 文件放置到主目录下后，即可通过 http://localhost/show_info.php 访问。如果 PHP 环境配置成功，会看到如图 5-14 所示的运行结果。

本例只用了一条语句来调用了 PHP 自带的 phpinfo 函数，输出 PHP 自身的配置信息。以此检验 PHP 环境是否搭建成功。如果无法显示图 5-14 中的结果，说明某个步骤出了问题，PHP 配置没有成功。值得注意的是，采用了 NTFS 文件系统的服务器，有时会因文件访问权限问题而导致 PHP 无法正常运行。因此，如果服务器采用了 NTFS 文件系统，至少

要保证 php.ini 和 PHP 安装目录以及 PHP 程序所在的目录都拥有足够的访问权限（推荐为 Everyone 可访问）。

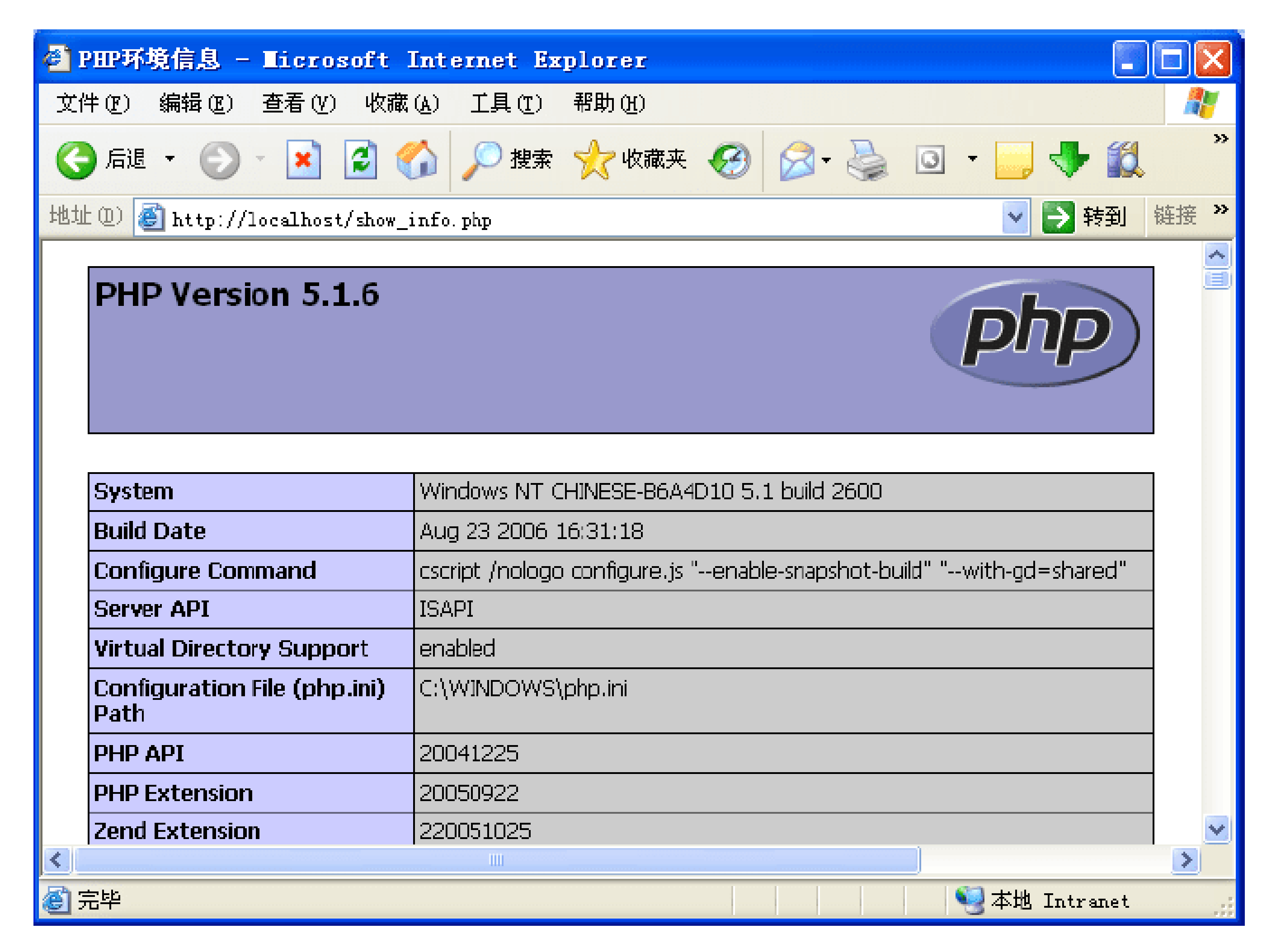


图 5-14　PHP 环境测试

5.7.2　Windows Server 2003 下安装 PHP

Windows Server 2003 系统（IIS 6.0 版本）下安装配置 PHP 的步骤与 Windows XP 下基本相同，简单介绍如下。

1. 安装 IIS 6.0

Windows Server 2003 在安装时会自动安装 IIS 6.0。如果系统没有安装，可以进行手工添加。添加步骤和 Windows XP 中基本相同，不同的是 Windows Server 2003 系统的“Windows 组件向导”界面中并没有直接列出“Internet 信息服务”项目。需要首先在组件列表中选择“应用程序服务器”，双击打开，出现应用程序服务器对话框，在子组件列表中将“Internet 信息服务（IIS）”复选框勾选，如图 5-15 所示。

单击“确定”按钮，然后单击“下一步”按钮，安装程序开始运行。程序运行结束后，IIS 6.0 安装完成。

2. 安装 PHP

安装 PHP 的步骤与 Windows XP 下步骤完全相同，因此略去，请读者参阅 2.3.1 节。

3. 将 PHP 与 IIS 建立关联

（1）按 2.3.1 节所述在 IIS 中添加 PHP 的应用程序影射。

（2）打开 IIS 管理器，在左侧目录树中找到“Web 服务扩展”项目，选择“添加一个新的 Web 服务扩展”命令，如图 5-16 所示。

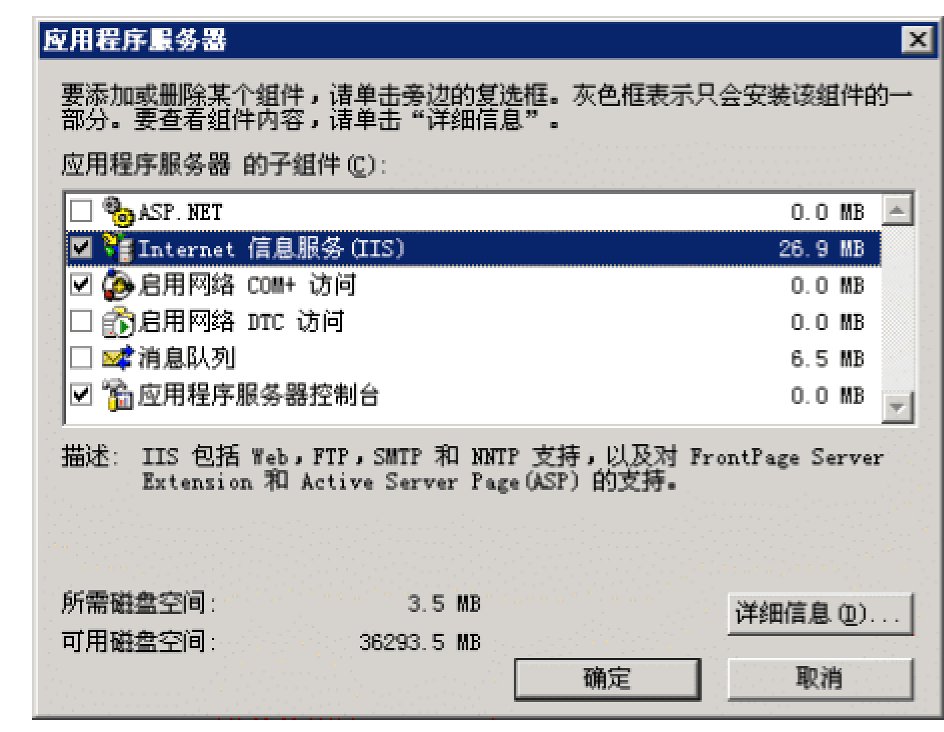


图 5-15　安装 IIS 6.0

在打开的“新建 Web 服务扩展”对话框的“扩展名”文本框中输入 php，选中“要求的文件”，单击“添加”按钮，选择 PHP 安装目录下的 PHP 5isapi.dll，并勾选“设置扩展状态为允许”复选框，如图 5-17 所示。

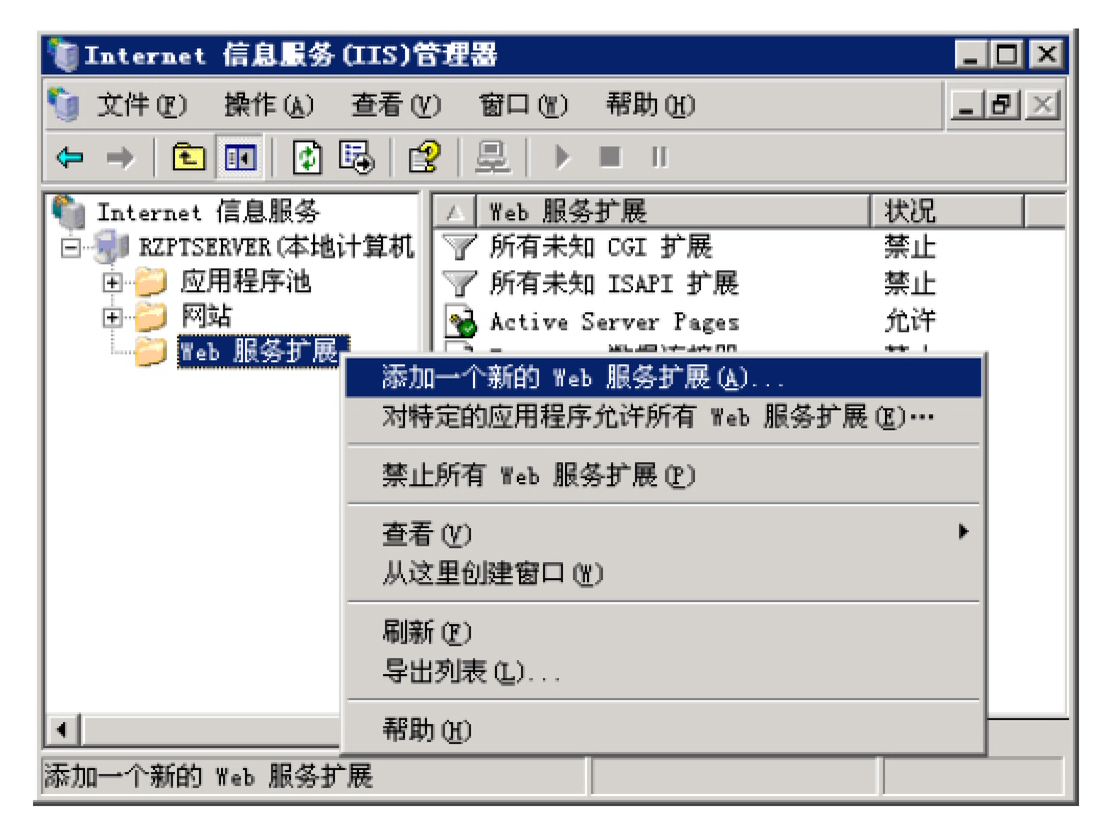




图 5-16　添加 Web 服务器扩展

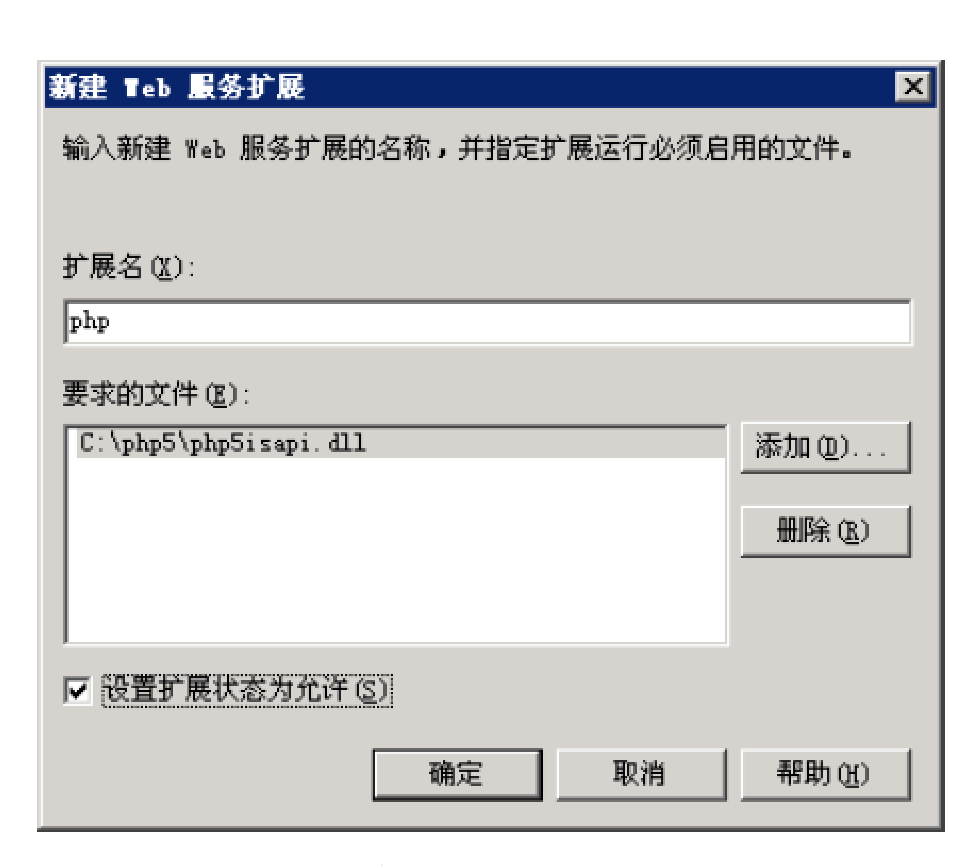


图 5-17　“新建 Web 服务扩展”对话框

单击“确定”按钮，可以看到右侧的“Web 服务扩展”列表中已经有了名为 php 且状况为“允许”的服务扩展。至此，PHP 与 IIS 的关联工作完成。

4. 运行测试

读者可按 2.3.1 节提供的方法编写简单程序测试配置是否成功。

5.7.3　IIS 主目录和虚拟目录设置

至止，基于 IIS 的 PHP 环境搭建工作已经基本完成。但是在正式编写 PHP 程序之前，有必要先学习一下 IIS 中如何设置主目录和虚拟目录。PHP 程序只有放在主目录或虚拟目录下才能调试、运行。

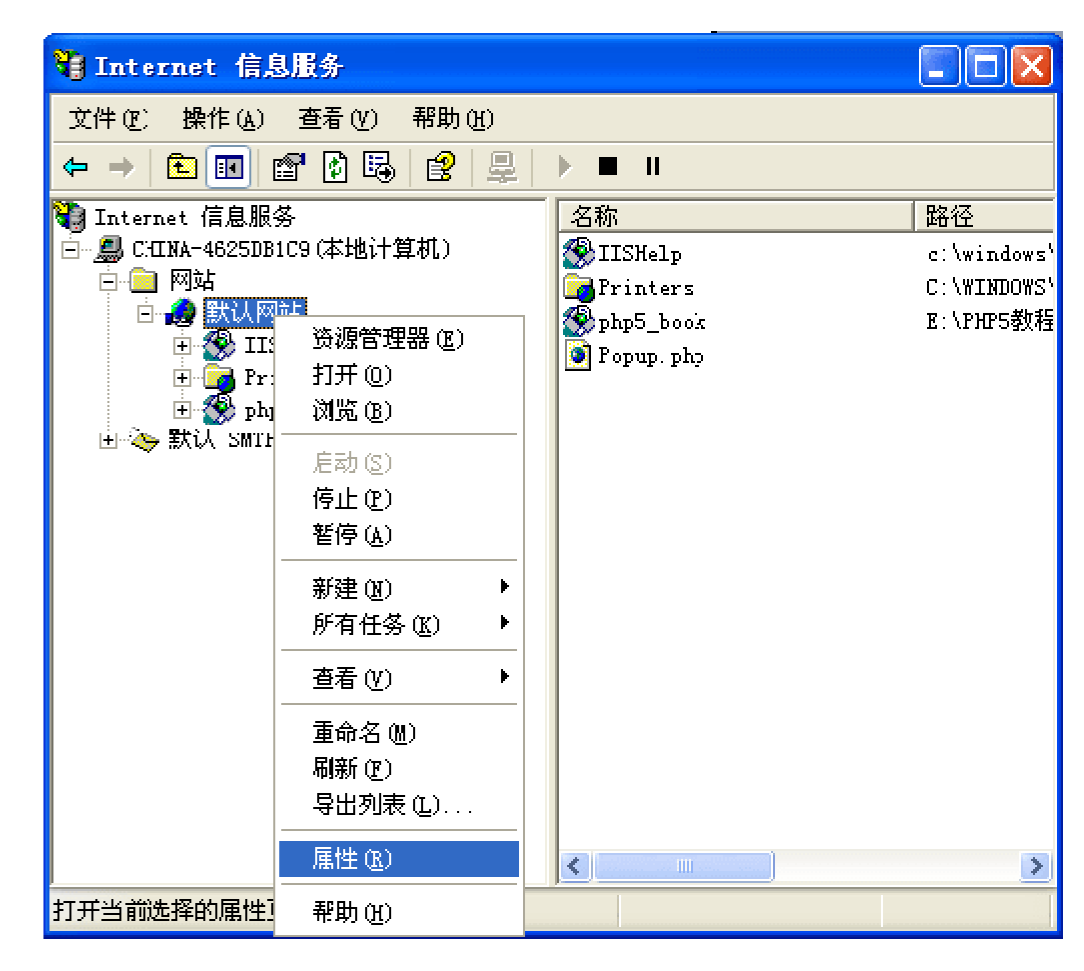
所谓主目录，就是服务器的默认站点在服务器上的存放位置。如某服务器 IP 地址为 10.0.0.10，当输入 http://10.0.0.10 这个地址访问网站时，服务器如何知道网站存放的位置呢？假设网站存放在 D:\wwwroot\目录下，只要在 IIS 中将主目录设置为 D:\wwwroot\，那么在访问 http://10.0.0.10 这个地址时 IIS 会自动到此目录下去搜索文件。如果将写好了的程序 index.php 存放在 D:\wwwroot\目录中，就只需要输入 http://10.0.0.10/index.php 即可看到该程序的运行结果。

在学习 PHP 阶段，一般都是以本地机器作为服务器，因此一般都输入 http://127.0.0.1 或 http://localhost 访问。在本书中的例子都是在本地服务器的主目录或者虚拟目录下运行的。

1. IIS 服务器主目录设置

以 IIS 5.1 为例介绍，其他版本的 IIS 也大同小异。打开 IIS 管理器，在左侧树形菜单中选择展开“网站”，在“默认网站”上右击，在弹出的快捷菜单中选择“属性”命令，如图 5-18 所示。

打开“默认网站 属性”对话框。单击“主目录”标签，如图 5-19 所示。



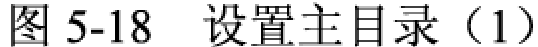

图 5-18 设置主目录（1）



图 5-19 设置主目录（2）

单击“浏览”按钮，选择计算机上的一个目录作为主目录。选择好之后依次单击“确定”按钮关闭所有对话框。这样 IIS 的主目录就设置完成。使用“http://localhost/文件名”这样的地址就可以访问此目录下的文件。如果主目录下还有子目录，就可以用“http://locahost/子目录名/文件名”访问了。

2. IIS 虚拟目录设置

有时，希望把一些网页单独存放在计算机上的某个目录下，而不是全部集中存放在网站主目录下，这时候可以采用虚拟目录。虚拟目录和主目录不同，一台服务器只能有一个主目录，但可以有多个虚拟目录。虚拟目录访问的方式是在服务器地址后面加一个虚拟目录名，如“http://localhost/虚拟目录名/”。

虚拟目录之所以称为“虚拟”，还有另外一层含义。那就是通过地址栏里的访问地址，无法确定网页的存放位置。例如，http://localhost/bbs/这个地址，无法确定/bbs 这个目录是主目录下面的一个子目录还是一个虚拟目录。虚拟目录可以存在于服务器上的任何位置，这就隐藏了实际的网页存储位置，有利于网站的安全。

创建一个虚拟目录的步骤如下。

（1）打开 IIS 管理器，展开“网站”，在“默认网站”上右击，选择“新建”→“虚拟目录”命令，打开“新建虚拟目录向导”对话框，如图 5-20 所示。

（2）单击“下一步”按钮，在“别名”文本框中输入该虚拟目录的名字（虚拟目录名任意，不必和文件夹名相同），假设取名为 ceshi，如图 5-21 所示。

（3）单击“下一步”按钮，出现网站目录选择页面。单击“浏览”按钮，选择一个目录作为该虚拟目录要访问的位置，也就是文件的存放位置。这里，假设选择 D:\wwwroot\。单击“下一步”按钮，出现“访问权限”页面，要求选择该虚拟目录允许进行的操作。对于一般的 PHP 站点，只选择前三项即可，即勾选“读取”、“运行脚本（如 ASP）”和“执行（如 ISAPI 应用程序或 CGI）”复选框，如图 5-22 所示。

（4）单击“下一步”按钮，单击“完成”按钮，即可完成虚拟目录的创建工作。创建完成之后会在左侧的树形菜单中出现虚拟目录名，如图 5-23 所示。



图 5-20　新建虚拟目录（1）

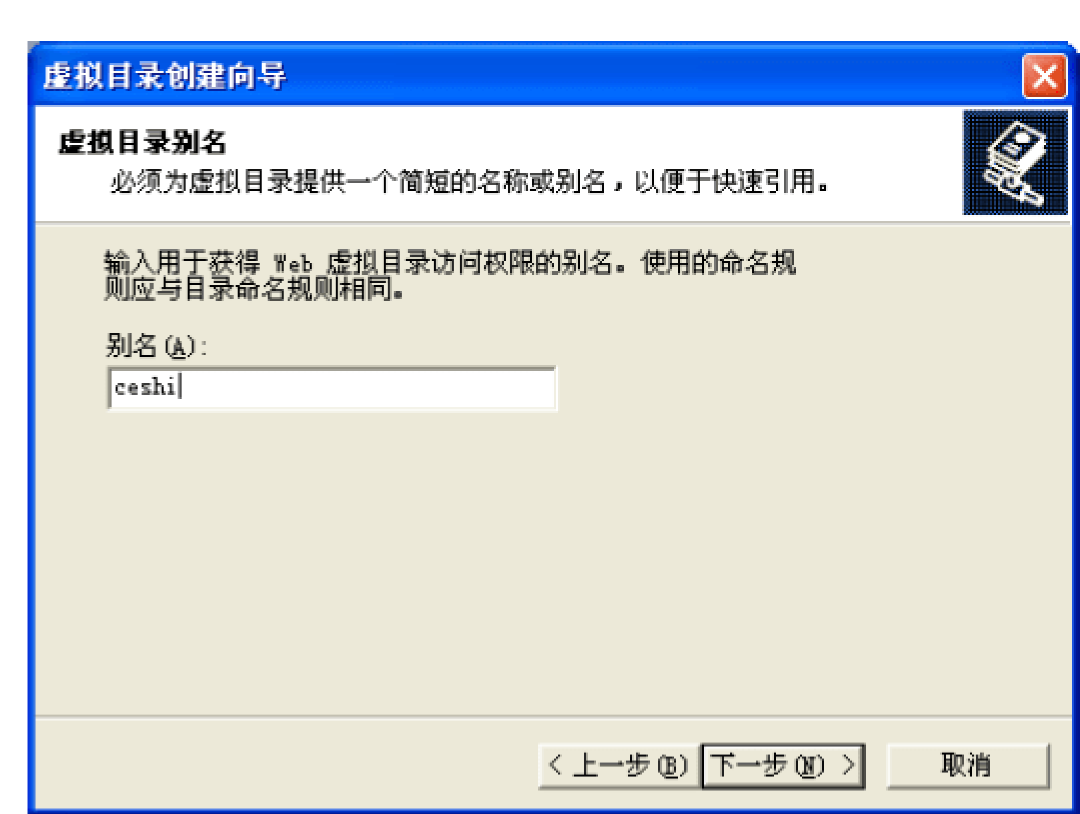


图 5-21　新建虚拟目录（2）

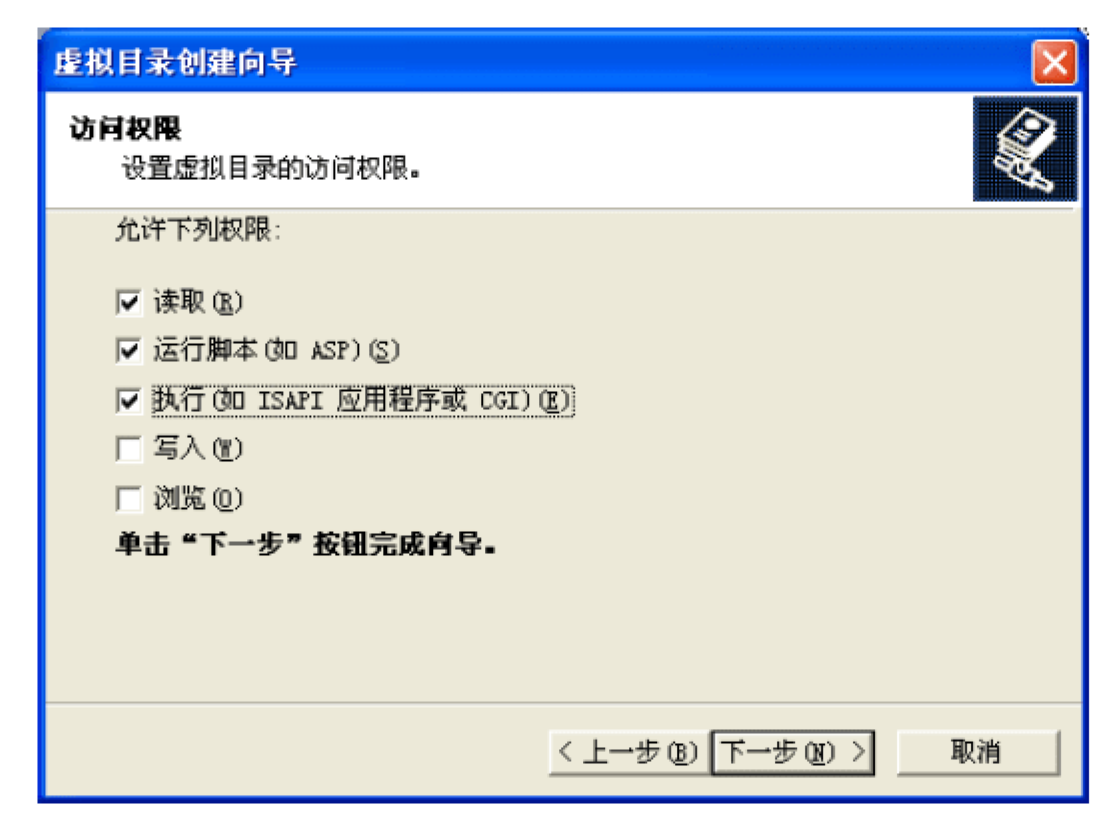


图 5-22　新建虚拟目录（3）

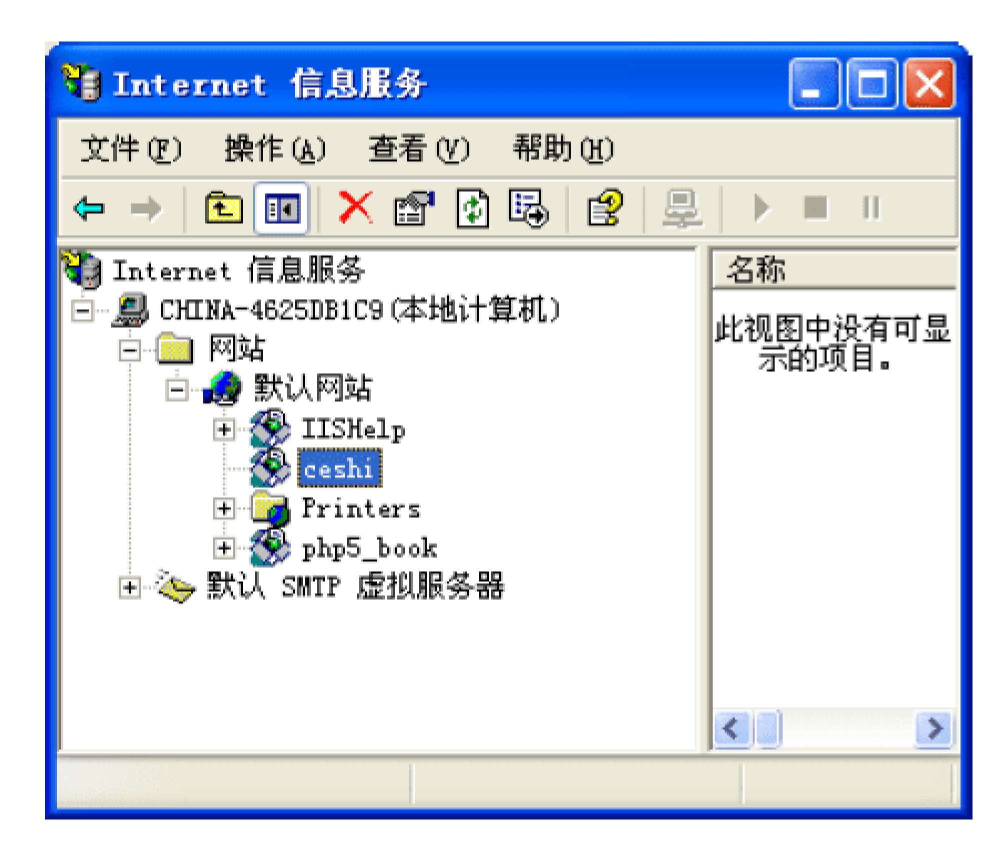


图 5-23　新建虚拟目录（4）

虚拟目录创建完成后，无论是 HTML 静态网页还是 PHP 程序，都可以放在 D:\wwwroot\目录下，然后用类似“http://localhost/ceshi/文件名”的地址访问即可。

另外，虚拟目录创建完成后，还可以对其属性进行设置和修改。方法与设置主目录相同，只需要在相应的虚拟目录的名称上右击，在弹出的快捷菜单中选择“属性”命令，在打开的窗口中进行操作即可。

上面简要介绍了 IIS 5.1 中设置主目录和虚拟目录的方法。在其他版本的 IIS 上设置的方法基本雷同，读者可以自行探索、试验，应灵活掌握 IIS 的基本操作和设置方法，才能更好地学习 PHP 编程和网站开发。

5.7.4　Windows + Apache 下安装 PHP

除了 IIS 以外，还有多款流行的 Web 服务器软件。Apache 就是其中最为著名的软件之一。事实上使用 Apache 来搭配 PHP 的服务器比 IIS 更为优越。另外，Apahce 还有一个重要的特性——跨平台。IIS 只能用在 Windows 操作系统上，而 Apahce 可以运行在包括 Windows 在内的许多主流操作系统上。

IIS 在实际使用中经常出现 500 错误，而且有的时候还会出现莫名其妙的假死现象。用

户需要不定期地重新启动 IIS 服务才能保证网站的正常。

Apache 在配置上比 IIS 要复杂，不过一经设置完毕就可以长期的工作了。大型网站都使用 Apache 作为自己的 WWW 服务提供工具。Apache 的所有配置都保存在配置文件中，使用时完全按照配置文件中记录的信息执行。一般不会发生莫名其妙的假死情况。

IIS 只能在微软公司的 Windows 操作系统下使用，离开了 Windows 将一事无成，无法移植到其他类型的操作系统中。

Apache 是个多面手，不仅仅应用于 Windows，对于 UNIX，Linux 以及 Freebsd 等多种操作系统来说都可以胜任工作，而且不同操作系统的配置步骤基本类似，可移植性非常高。

早期的 IIS 在安全性方面存在着很大的问题，如果使用默认设置，黑客可以轻松趁虚而入。在 IIS 6.0 中微软公司对安全方面进行了大幅改进，只要保证操作系统补丁更新及时，就可以将网站安全系数尽可能的提高。特别是 IIS 6.0 与.Net 平台相互倚靠，使安全性几乎完美。

Apache 在安全方面一直做的很好，因为很多用户都是在 Linux 下使用 Apache，所以操作系统的特点使得 Linux 下的 Apache 具有先天的保护伞。

所谓开放性是指是否开放了程序的源代码，众所周知 IIS 是 Windows 系统的一部分，它的源代码是没有开放的。而 Apache 则不同，最早它是为了类 UNIX 系统服务的，所以完全对外开放源代码。任何人都可以分析他的代码，发现其中的漏洞，并发布补丁弥补该漏洞。

因为 Apache 的这种开放性，也使其安全性大大提高。

IIS 对 ASP 特别是.Net 运行很稳定，不过对于 PHP 和 JSP 就比较麻烦了。PHP 需要经过反复配置才能在 Windows 2003 上支持。Apache 则能够很好地支持上面提到的几种语言，运行 ASP、PHP、JSP 都没有任何问题。

Apache 是目前世界上使用最为广泛的 Web 服务器之一，根据 NetCraft 公司的调查，世界上 50%以上的 Web 服务器都在使用 Apache。

和其他服务器相比，Apache 拥有以下主要特性。

（1）几乎可以运行在所有的计算机平台上。

（2）支持最新的 http/1.1 协议。

（3）简单而且强大的基于文件的配置（httpd.conf）。

（4）支持通用网关接口（CGI）、FastCGI、支持虚拟主机、支持 http 认证。

（5）具有用户会话过程的跟踪能力。

（6）支持 Java Servlets。

（7）运行效率高，成本低。

当前，Apache 已经发展到 2.2.3 版本。不过常用的 Apache 版本有 Apache 1.3 和 Apache 2.0.x。由于 PHP 和 Apahce 的不同版本之间存在兼容性问题，因此在选择 Apache 的版本时应当同时考虑 PHP 的版本。在本书中使用的 PHP 为当前最新的 PHP 5.1.6，可以使用 Apache 2.0.x 版本与之搭配。

了解了 Apache 的概况后，就来看 Apache 下 PHP 的安装配置。

1. Apache 的获取

由于 Apache 也是免费软件，因此有很多途径可以获取 Apache 的安装文件。最稳妥的

方法当然还是从其官方网站直接下载。Apache 的官方网站是 http://www.Apache.org。

打开官方站点首页，如图 5-24 所示，在左侧的产品分类中选择 HTTP Server 链接，在打开的网页中选择合适的 Apahce 版本，单击 Download 项，即可进入下载页面。单击相应下载链接即可获得 Apache 安装文件，如图 5-25 所示。

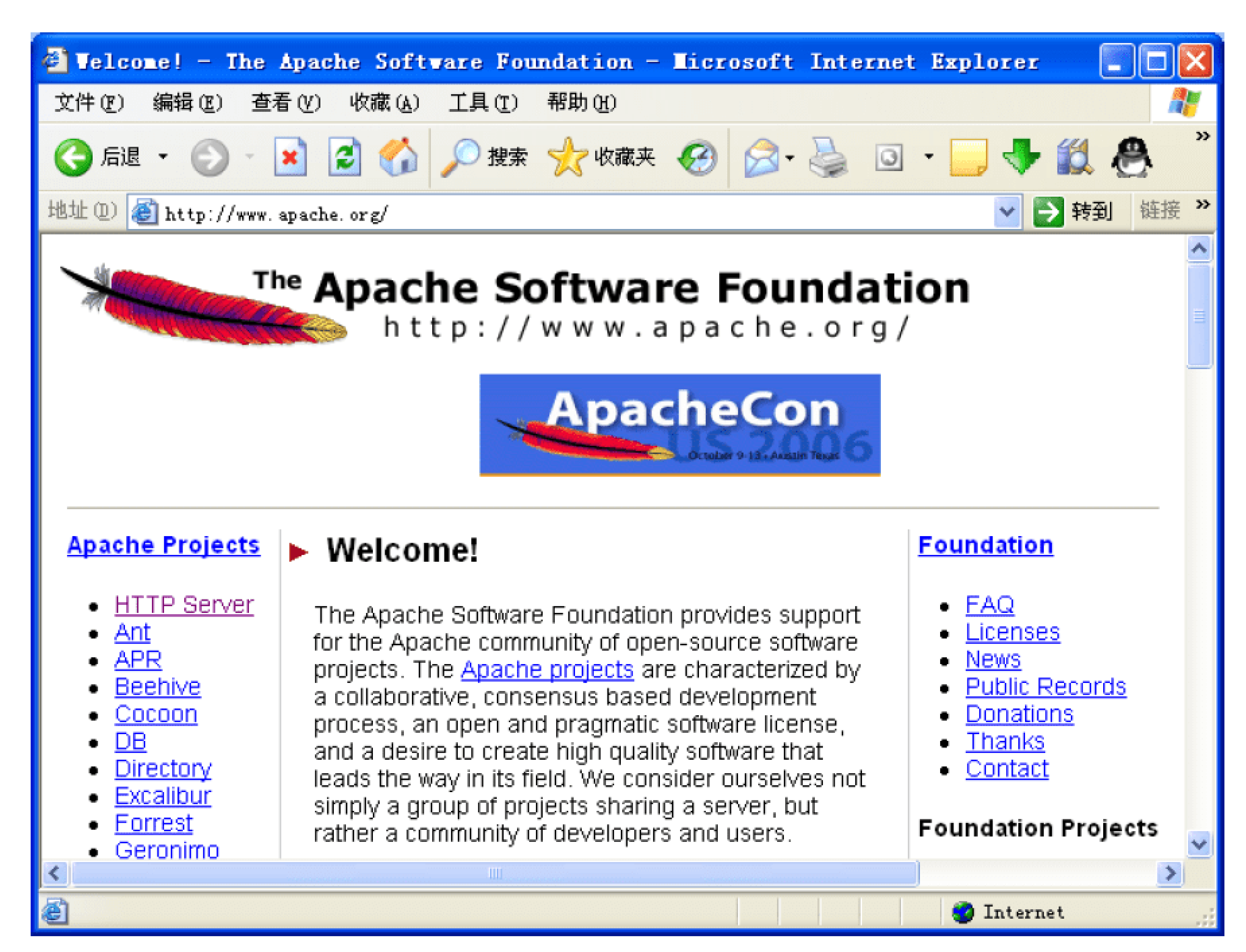


图 5-24 Apache 官方网站

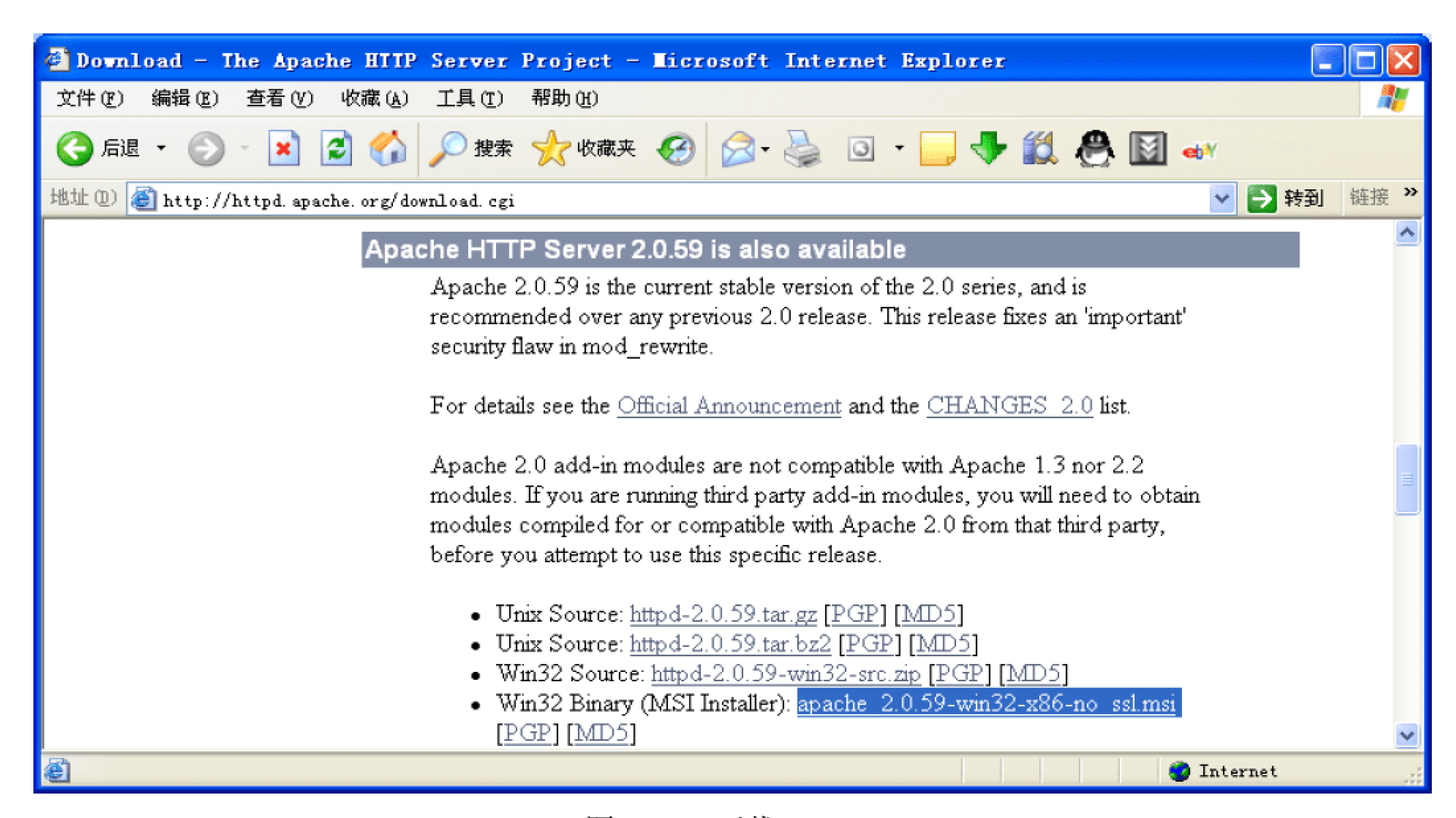


图 5-25 下载 Apache

这里下载的是 Apache 2.0.59，文件名为 Apache_2.0.59-win32-x86-no_ssl.msi，这是一个可以双击直接运行的安装程序。它属于 Apache 2.0.x 系列，能够运行于 Widnows 系统平台，并与 PHP 5 搭配使用。

2. Apache 的安装

Apache 在 Windows 平台上的安装非常简单，只需要启动安装程序，按照提示逐步进行即可。不过在安装 Apache 之前，必须保证计算机没有安装 IIS 或其他服务器软件，如果已经安装了，建议将其卸载。因为可能会因端口冲突而导致 Apache 无法启动。当然可以通过一些设置使各个服务器软件工作在不同的端口，但是在绝大多数情况下只需要一个服务器软件就足够了。

双击启动 Apache 安装程序，出现软件的欢迎界面，单击 Next 按钮，出现 Apache 许可协议，阅读完毕许可协议之后选择 I accept the terms in the license agreement 复选框，表示接受许可协议中的条款。再单击 Next 按钮，出现 Apache 服务器注意事项，阅读完毕后单击 Next 按钮，出现服务器信息设置界面，如图 5-26 所示。此处要求设置一些服务器基本信息，分别是网络域名、服务器名、管理员信箱以及 Apache 的工作方式。如果只在本地机器上使用 Apache，前 3 个选项可以保持空白，不需要设置。第 4 个选项有两种选择，建议选择第一项，也就是“针对所有用户，工作在 80 端口，安装为服务”。“安装为服务”的意思是将 Apache 安装为 Windows 的一个服务，当计算机启动时自动启动 Apache。

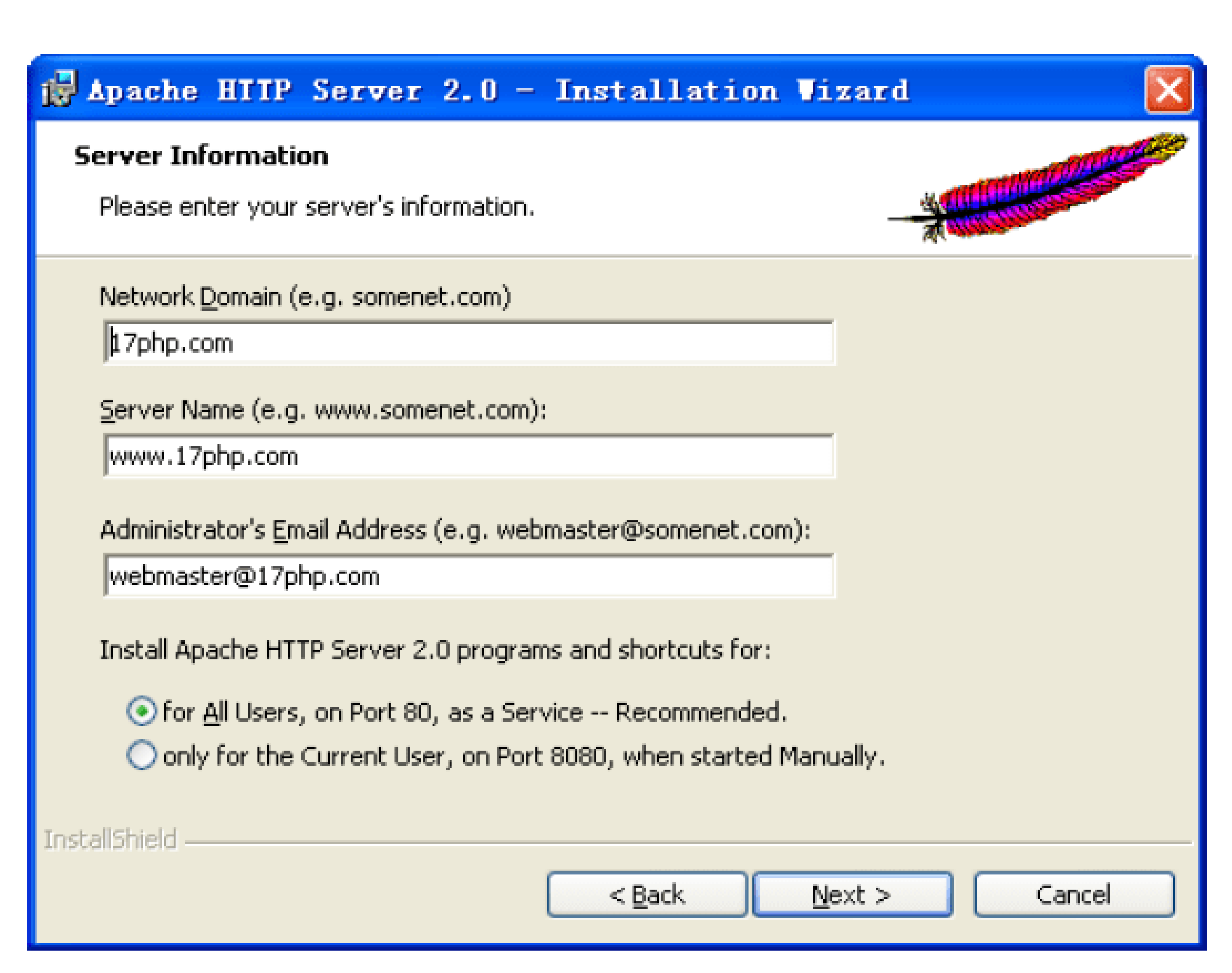


图 5-26 Apache 服务器信息设置

设置完成之后，单击 Next 按钮进入下一步。这时出现安装类型窗口，在这里可以选择 Typical 或 Custom 单选按钮，即“典型安装”和“用户自定义安装”两种选择，对于对 Apache 不太熟悉的初学者，建议直接使用“典型安装”，如图 5-27 所示。

继续单击 Next 按钮，出现 Apache 安装位置选择窗口。Apache 默认被安装到 C:\Program Files\Apache Group\目录下。如果希望安装在其他位置，可以单击 Change 按钮来选择另外一个位置。这里采用默认位置，单击 Next 按钮，出现“安装准备已就绪”窗口。如果不需要对前面进行过的步骤做修改，就可以直接单击 Install 按钮开始安装 Apache。安装开始后会出现安装进度条，如图 5-28 所示。

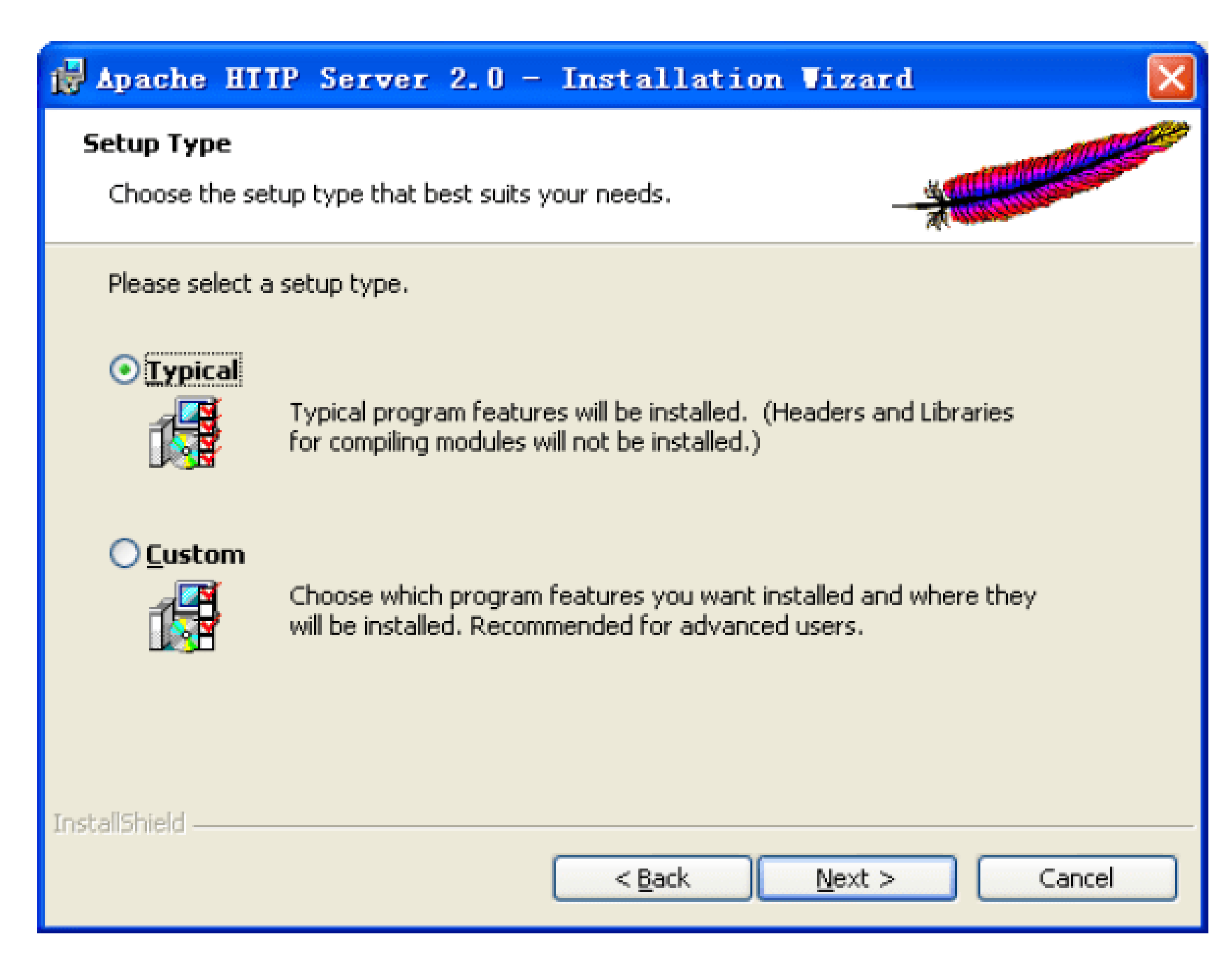


图 5-27　选择安装类型

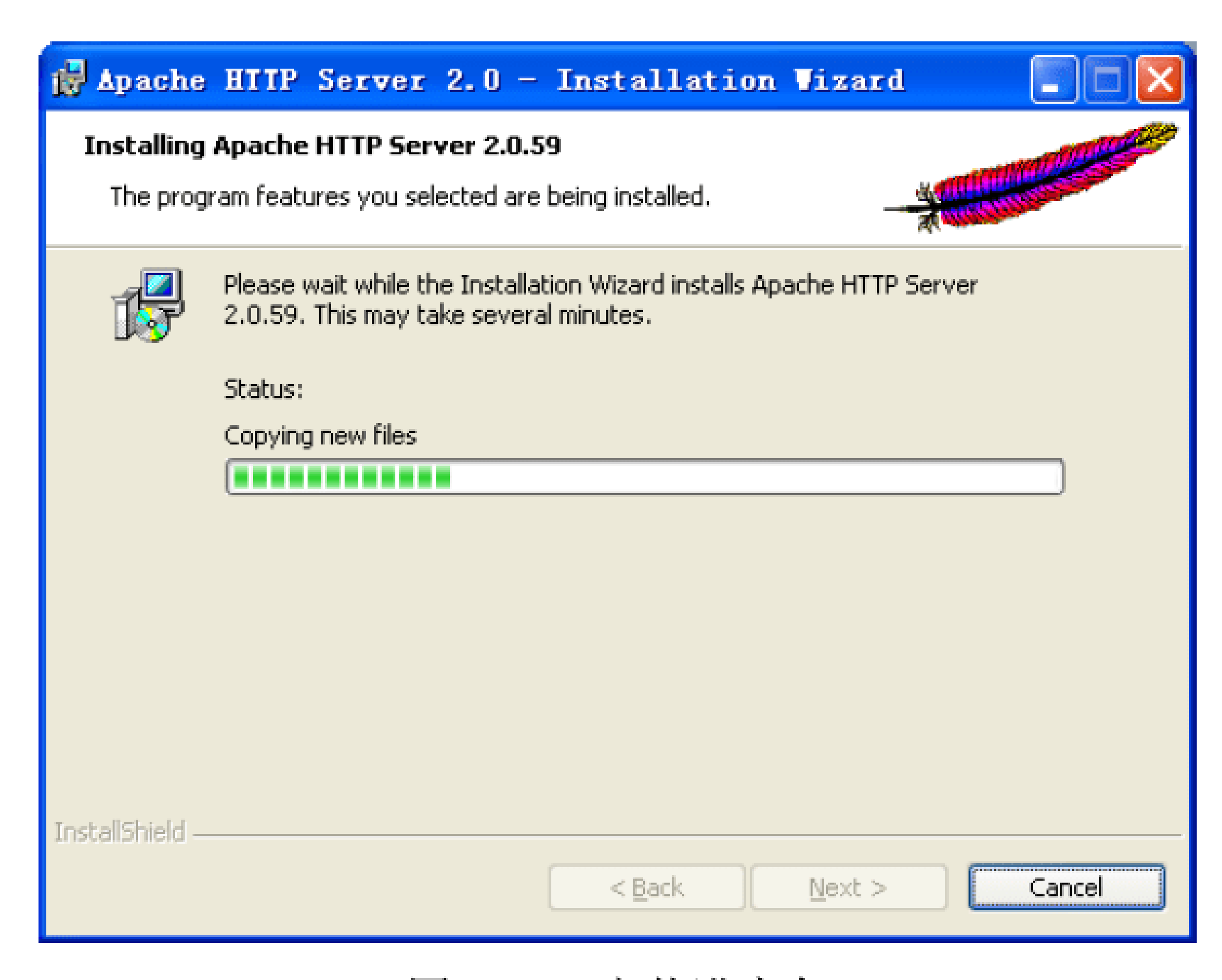


图 5-28　安装进度条

全部安装进行完成后，会出现安装成功的提示窗口，单击 Finish 按钮结束安装程序，Apache 的安装完成了。为了验证安装是否成功，打开浏览器，在地址栏中输入 http://localhost/或 http://127.0.0.1/，如果看到如图 5-29 所示的页面，那么说明 Apache 已经成功安装并开始服务了。

安装成功后在系统程序组中会出现一个名为 Apahce Http Server 2.0.59 的程序组，其中包含了对 Apache 服务器的一些控制功能，如配置 Apahce 参数、启动、关闭、重新启动 Apache 服务以及帮助信息等。

最后还要强调一点：虽然绝大多数情况下都可以快速顺利地安装成功 Apache，但也不排除安装失败的情况。由于操作系统版本、计算机软件环境等影响，有可能安装 Apache 的最后阶段会出现错误，或安装之后无法启动。这时应根据 Apache 给出的错误提示考虑出错的原因。常见的错误有找不到 Apache2 服务、端口冲突等。

对于找不到 Apache2 服务，说明 Apache 没有成功地被安装为 Windows 服务，这时候可以手工启动 Apache，也可以在命令行模式下将其注册为服务。端口冲突一般是由于安装

了其他软件占用了 80 端口所致，可以通过卸载无关软件或者修改 Apache 服务端口的办法解决。如果计算机安装了防火墙等软件，也有可能导致 Apache 因无法打开端口而无法启动。总之，如果遇到安装问题，可以通过阅读 Apache 的安装说明文档或者上网检索寻求解决办法。



图 5-29 Apache 安装成功

3. 安装 PHP

PHP 的安装步骤与 2.3.1 节中的 PHP 安装步骤完全相同，也是解压、修改 php.ini、复制文件等步骤，请读者参阅该节进行基本安装。

4. 将 PHP 与 Apache 建立关联

虽然 Apache 目前已经可以正常运行，并能提供静态网页服务，但此时它仍无法运行 PHP 网页。同 IIS 一样，需要先告诉 Apache 如何处理 PHP 程序。想让 Apache 能够运行 PHP 程序，也必须将 PHP 与 Apache 建立关联。

（1）找到 Apache 配置文件。同 IIS 的图形界面不同的是，Apache 的设置是通过修改其配置文件来实现的。这个配置文件的名字为 httpd.conf，是一个纯文本文件，可以直接用记事本程序打开并编辑。这个文件被存放在 Apache 安装目录的 Apache2\conf\目录下，也可以通过 Apache 程序组中的 Configure Apache Server 子程序组下面的 Edit the Apache httpd.conf configuration file 菜单直接打开这个配置文件，如图 5-30 所示。

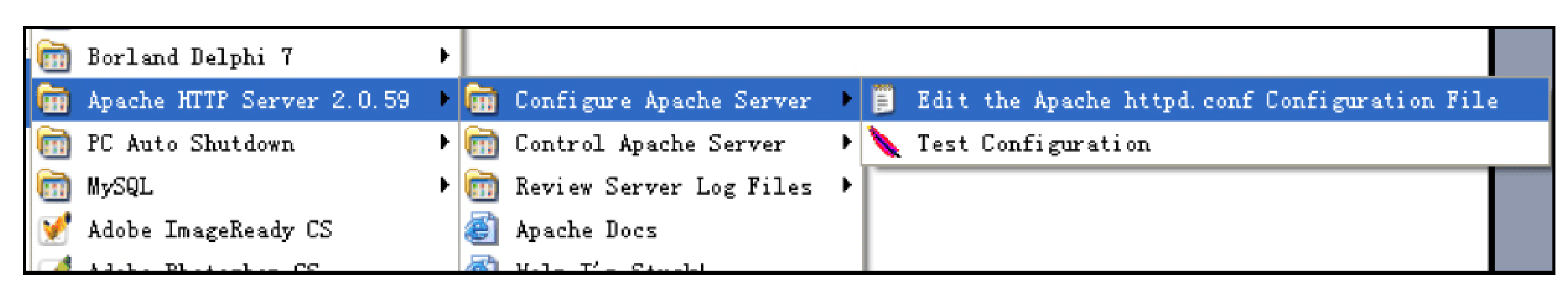


图 5-30 打开 Apache 配置文件

（2）修改 Apache 主目录。打开 httpd.conf 之后，首先要做的就是设置网站的主目录，也就是在默认情况下网页存放的位置。默认 Apache 安装目录为 Apache2\htdocs\。如果把 Apache 安装在 C:\Program files\Apache Group\，那么默认的主目录就是 C:\Program Files\Apache Group\Apache2\htdocs\。可以修改这个目录来适应自己的需要。如在 D 盘创建一个目录 wwwroot，可以通过以下方法将此目录设置为主目录。

在 httpd.conf 中找到 DocumentRoot 参数，将其值修改为 D:/wwwroot/，如图 5-31 所示。可以看出，Apache 的配置文件与 PHP 的形式上有所不同。PHP 配置文件中以 “=” 作为参数名和参数值的分隔符，而 Apache 中使用空格。要特别注意的是，PHP 配置文件中表示路径时目录分隔符用 “\”，而 Apache 中用 “/”。



图 5-31　修改 Apache 的主目录

（3）向 Apache 中加入 PHP 模块。有两种方式可以将 PHP 与 Apache 建立关联，一种是 CGI 方式；另一种是模块方式。这里采用的是模块方式。在 httpd.conf 中任意位置插入以下两行代码：

```
LoadModule  PHP 5_module  "c:/PHP 5/PHP 5Apache2.dll"
AddType  application/x-httpd-php  .php
```

第一行代码的作用是使得 Apache 在启动时载入 PHP 模块。第二行代码的作用是使 Apache 能够识别 php 的扩展名。必须注意的是，第一条语句中的 PHP 5Apache2.dll 容易错写成 PHP 5Apache.dll，因为在 PHP 的安装目录下这两个文件都存在。要使用 PHP 5Apache2.dll 是因为采用的 Apache 版本为 2.0.59，属于 Apache 2.0.x 系列。如果使用的 Apache 版本是 1.3.x，那么此处就应该使用 PHP 5Apache.dll。

（4）指定 php.ini 文件的存放路径。Apache 中还有另外一个重要参数用来指定 php.ini 文件的存放位置。由于 PHP 安装时将 php.ini 复制到 C:\WINDOWS\中，因此其位置就是 C:\WINDOWS\php.ini。在 httpd.conf 中任意位置加入下面一条语句：

```
PHPIniDir "C:/WINDOWS/"
```

或

```
PHPIniDir "C:/WINDOWS/php.ini"
```

这样，PHP 便知道到哪里去搜索 php.ini，这样 php.ini 中的设置才能生效。添加完成后如图 5-32 所示。

上面 4 个步骤完成后，保存 httpd.conf 文件，重新启动 Apache 使设置生效。重新启动的办法是在 Apache 程序组的 Control Apache Server 中选择 Restart，稍后 Apache 启动。PHP 和 Apache 的关联完成。

下面可以编写一个例子，测试 Apache 服务器是否可以正常运行 PHP 程序。若使用本章 2.3.1 节中的测试代码，将其保存到主目录 D:\wwwroot 中，然后在浏览器中输入 http://localhost/show_info.php，这时应该可以看到与图 5-14 几乎完全相同的输出结果，只不过表中的个别

参数与使用 IIS 时略有不同，表明 Apahce2＋PHP 5 服务器环境搭建成功。



图 5-32 修改 httpd.conf

5.8 Linux 下 PHP 的安装与配置

Linux 在当今服务器操作系统领域占有重要的地位，PHP 可以完美地运行在 Linux 平台上，因此介绍 Linux 下的 PHP 安装配置就显得很有必要。

与 Windows 不同，Linux 是一个开源的系统，其版本比较多，名称也各不相同，如 Red Hat Linux、SuSe Linux、Ubuntu Linux、Debian Linux、红旗 Linux 等。

在很多 Linux 版本上，PHP 的安装步骤略显复杂。因为要用到一些 Linux 命令对 PHP 进行解压、编译、配置。根据 Linux 的版本不同，其步骤又各有区别，这就很大程度上增加了 Linux 下配置 PHP 的难度，对于初学者来说不易掌握。不过随着技术的发展，现在已经有 Linux 版本支持通过直观、易操作的方式在 Linux 上配置 PHP＋MySQL＋Apache。如 Red Hat 发布的 Fedora 8，在 Fedora 8 中直接集成了开发 PHP 所需要的 PHP 安装包、Apache 服务器、MySQL 数据库和其他工具。只需要在安装 Fedora 时选配这些组件，系统装好之后即可以轻松开启 PHP 支持。相当于省略了手工安装步骤，只需要设置服务器是否开启或关闭 PHP 支持即可，非常方便。

Fedora 是基于 Linux 的操作系统，包含了自由和开源软件最新的成果。和绝大多数 Linux 系统一样，Fedora 允许所有人自由使用、修改和重新发布。可以通过 Fedora 项目的官方网站直接下载 Fedora 的安装文件（一般为 ISO 光盘映像）。

介绍 Linux 并非本书的主要内容，因此关于 Linux 的历史、下载安装以及使用知识请读者参阅其他相关书籍。本书以 Fedora 8 为例，简要介绍如何在 Fedora 上运行 PHP。

1. 选配 Fedora 组件

在 Fedora 安装中的额外功能选项步骤默认的只有“办公”。勾选“网络服务器”复选框，并选择下方的“现在定制”单选按钮，然后单击“下一步”按钮，如图 5-33 所示。

进入下一步后，可以自由定制需要安装的服务器软件。在出现的窗口的左侧选择“服务器”项，右侧列出了服务类型，其中包含了许多常见的服务。出于学习 PHP 的需要，建议勾选“MySQL 数据库”、“万维网服务器”复选框。为了更进一步选择服务器的组件，可以单击下方的“可选的软件包”按钮，如图 5-34 所示。

打开“可选的软件包”后，可以看到所列出的软件包，其中已经包含了 Apache、PHP 等，如图 5-35 所示。一般情况下如果没有特殊需求，不需要对其进行修改，按照默认设置即可。



图 5-33　Fedora 安装

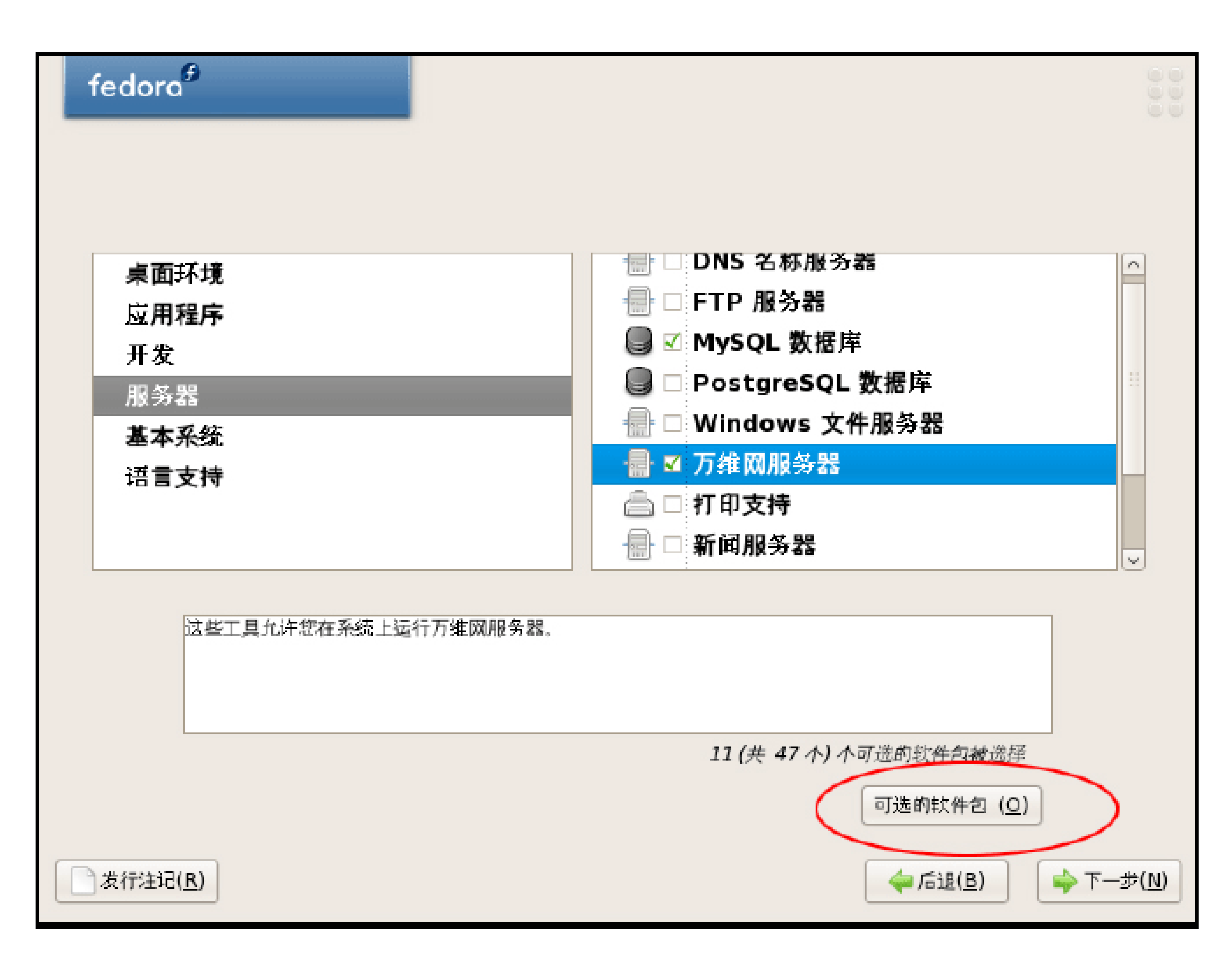


图 5-34　定制服务器组件



图 5-35　可选软件包的自定义

一切设置好后，可以进入下一步，按提示安装完成 Fedora。

2. 开启 Apache 服务

Fedora 安装完成后，默认情况下 Apache 服务是关闭的，需要进行手工开启。开机进入 Fedora 桌面，选择“系统”→“管理”→“服务”命令，如图 5-36 所示。

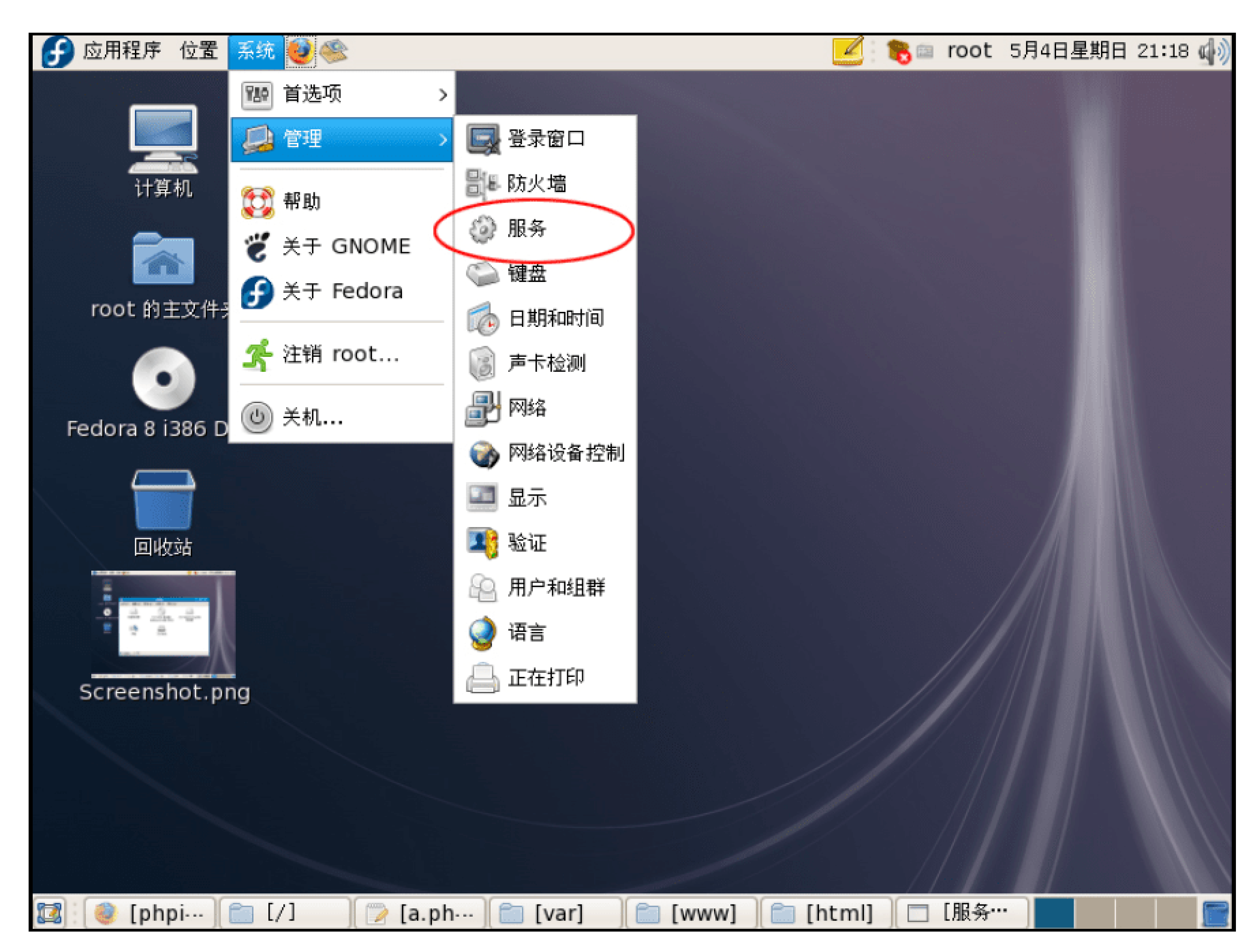


图 5-36 开启 Apache 服务

选择执行“服务”之后可以打开“服务配置”对话框。在对话框中找到 httpd 项，勾选此项并单击“开始”，稍等之后 Apache 即可启动成功，如图 5-37 所示。



图 5-37 Fedora 服务配置

Apache 启动之后，可以打开浏览器测试一下是否能够访问本地服务器。打开 Fedora 自带的 FireFox 浏览器，在地址栏中输入 http://localhost，可以看到如图 5-38 所示的界面，说明 Apache 服务启动成功。

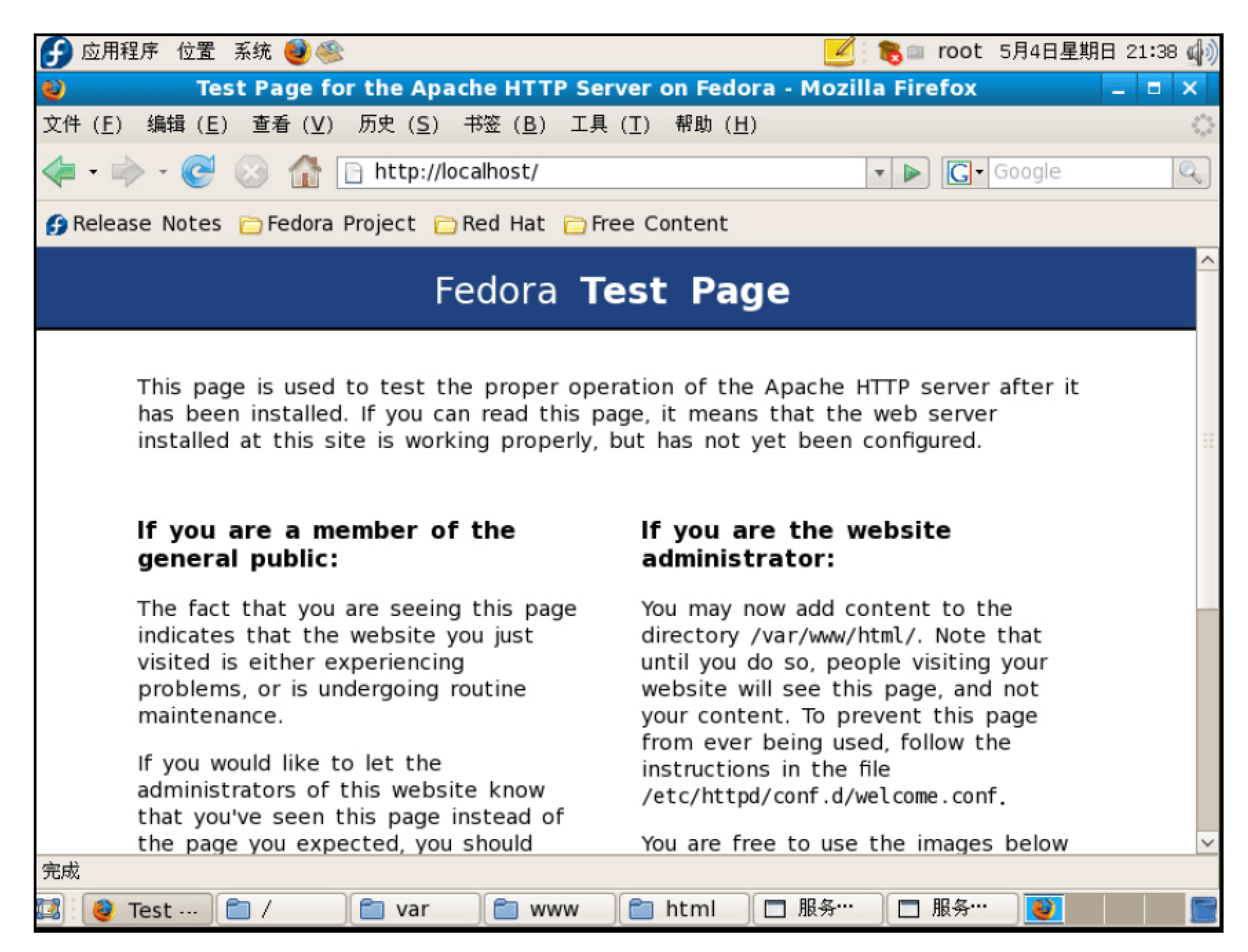


图 5-38　Fedora 服务器默认首页

3. 运行 PHP

实际上，Fedora 已经安装好了 PHP，而且一般都是最新版本。几乎不需要自己做任何配置，就能直接使用 PHP 了。再用 2.3 节中的测试程序来检查 Fedora 服务器是否已经能够支持 PHP，同时可以了解 PHP 的版本。

```
<?php
   phpinfo();
?>
```

可将此文件保存为 test.php，并将其复制到 Apache 主目录下（Fedora 默认为/var/www/html/），然后在地址栏中输入 http://localhost/test.php，可以看到成功输出了 PHP 配置信息，证明 PHP 运行成功。运行结果如图 5-39 所示。

至此，在 Fedora 上安装 Apache+PHP 的步骤已经完成。可以看出，Fedora 提供了完美的 PHP 支持，开发者可以轻松地使用 PHP。由于 Fedora 内置 PHP 支持，使得在 Fedora 下配置 PHP 比 Windows 下都要简单许多。

当然，Fedora 自动安装的 Apache 和 PHP 都是按照默认配置进行设置的，为了满足用户个性化的需要，可以对其进行进一步自定义配置，如修改 Apache 主目录、添加虚拟目录、站点以及修改 PHP 配置信息等。这部分内容请感兴趣的读者参考有关资料自行实现。

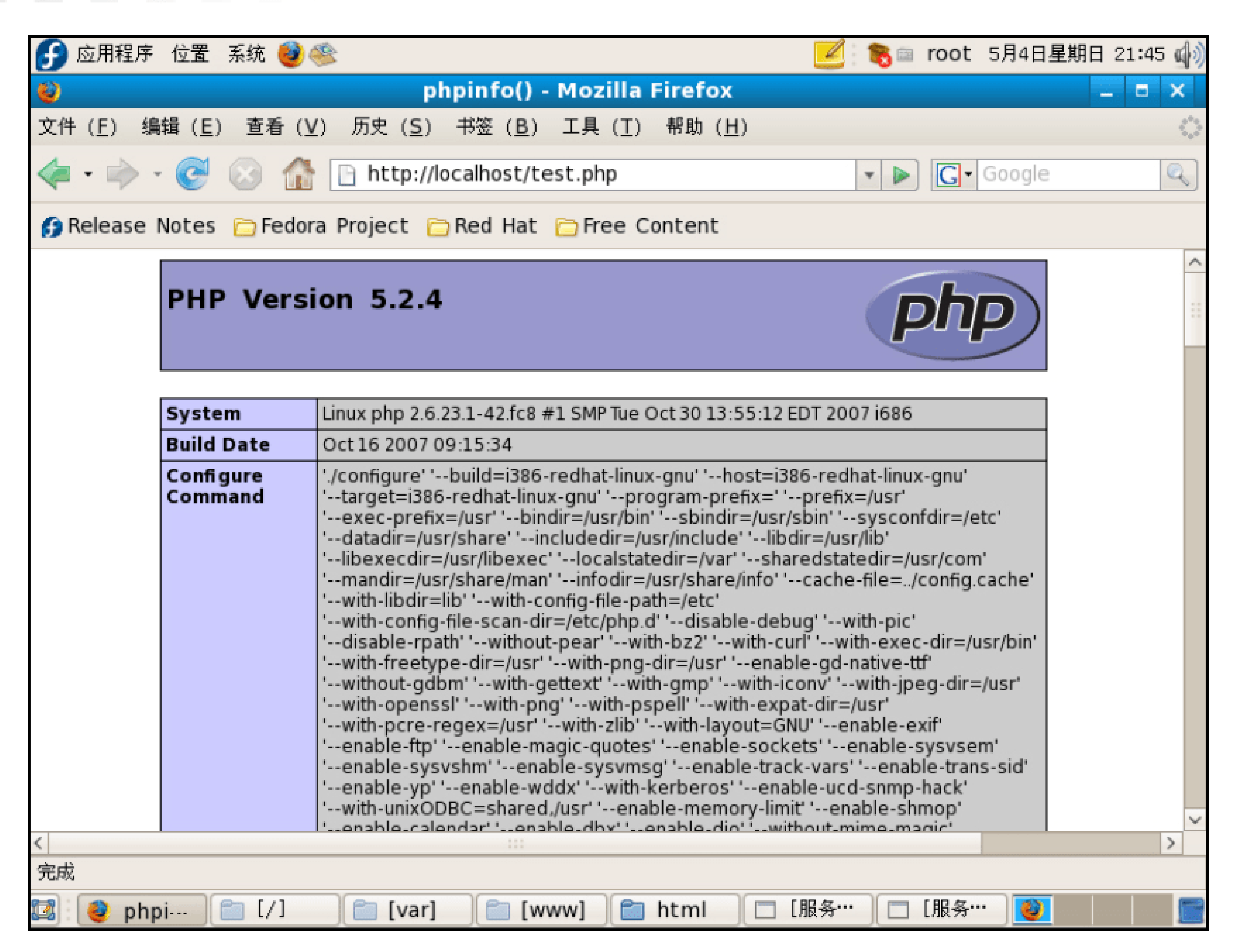


图 5-39 PHP 测试程序

5.9 本章小结

本章介绍了 PHP 的来历和应用领域，介绍了 PHP 的主要特点和常用的几种开发工具。对读者而言，成功地搭建 PHP 的运行环境，是学习 PHP 的基础，因此本章分别对 Windows 和 Linux 两种主要操作系统下如何搭建 PHP 的运行环境进行了详细说明，读者可根据喜好选择其中的一种。“工欲善其事，必先利其器”，读者应当根据本章的讲解，多次练习，反复尝试。通过本章的学习，读者应能熟练掌握 PHP 运行环境的配置和 Web 服务器的配置方法，从而为 PHP 的学习做好充分地准备。

5.10 练习题

1．什么是 PHP？简述 PHP 的发展历史。
2．PHP 有哪些特性？你准备选择哪种 PHP 开发工具？
3．简述 PHP 的运行流程。
4．简述 IIS Web 服务器和 PHP 的关联方法。
5．简述 Apache Web 服务器和 PHP 的关联方法。
6．什么是虚拟目录？简述主目录下的子目录和虚拟目录的不同之处。
7．到网上下载合适的 PHP 版本，按步骤完成 PHP 的安装。
8. 分别下载 IIS 和 Apache 两种 Web 服务器，按步骤完成安装，并建立与 PHP 的关联。

第6章

PHP 5 的基本语法

本章重点：

- PHP 程序规范；
- PHP 的数据类型；
- PHP 中的变量和常量；
- PHP 变量的作用域；
- PHP 运算符和表达式；
- PHP 流程控制语句；
- PHP 网页间通信。

6.1 PHP 程序规范

6.1.1 第一个 PHP 程序

在第 2 章中，已经搭建了 PHP 的运行平台，下面就来学习 PHP 的基本语法。“hello, world!”几乎已经成了所有程序语言的第一个范例，本节也不例外，先写一个输出“hello, world!”的简单的 PHP 程序。

程序 6-1.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-1.php：第一个 php 程序-->
      <head>
           <title>6-1.php</title>
      </head>
      <body>
          <?php
             echo "hello, world!";
          ?>
     </body>
</html>
```

这个程序在 PHP 中不需经过编译等复杂的过程，只要将它放在已配置好 PHP 平台的服务器中，并以 6-1.PHP 文件名保存此程序。在用户端的浏览器中，在地址栏中输入 http://localhost/phpsource/6/6-1.php（phpsource 为存放 php 文件的文件夹的服务虚拟目录，6 为该目录下的一个文件夹，6-1.php 为 6 文件夹下的一个 PHP 文件），就可以在浏览器上看到如图 6-1 所示的效果。



图 6-1　程序 6-1.php 的运行结果

可以看到，这个程序中 PHP 代码只有 3 行，其他行都是标准的 XHTML 语法。第 9 和第 11 行分别是 PHP 的开始和结束的嵌入符号。第 10 行才是服务器端执行的语句。可以通过选择浏览器窗口的“查看”→“源文件”命令来查看其源文件。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-1.php：第一个 php 程序-->
  <head>
      <title>6-1.php</title>
  </head>
  <body>
      hello, world!
</body>
</html>
```

PHP 程序在返回浏览器时，和 JavaScript 或 VBScript 完全不一样，PHP 的源程序没有传到浏览器，只在浏览器上看到“hello, world!”。

6.1.2　PHP 代码的嵌入方式

在文件 6-1.php 中“<?php”和“?>”为 PHP 的分界符，表示其中所包含的代码是 PHP 代码，当服务器读到该段程序的时候就会调用 PHP 的编译程序。将 PHP 代码嵌入到 XHTML 中共有 4 种方式，具体如下。

1. 利用分界符“<?php”和“?>”

这是 PHP 最为普通的嵌入方式，也是 PHP 标准的嵌入方式，举例如下：

```
<?php
echo（"这是一个标准方式的 PHP 语言的嵌入范例"）;
?>
```

推荐使用此种嵌入方式，这种写法可以为程序在跨平台使用时减少不必要的麻烦。

2. 利用分界符“<?”和“?>”

这种方式是简写方式，必须在 php.ini 文件中将 shor_open_tag 设置为 On（PHP 5 中默认设置为 On）；否则编译器将不予解析。例如：

```
<?
echo（"这是一个简写方式的 PHP 语言嵌入范例"）;
?>
```

3. 利用分界符“<script language="php">”和“</script>”

这是类似于 JavaScript 和 VBScript 风格的嵌入方式，对熟悉 Netscape 服务器产品的人员而言，应该相当亲切。例如：

```
<script language="php">
echo ("这是类似 JavaScript 和 VBScript 风格的 PHP 语言嵌入范例");
</script>
```

4. 利用分界符“<%”和“%>”

这是一种具有 ASP 风格的嵌入方式，必须在 php.ini 文件中设置 asp_tags 为 On，否则编译器将不予解析。例如：

```
<%
echo ("这是类似 ASP 风格的 PHP 语言嵌入范例");
%>
```

建议少用这种方式，因为 PHP 与 ASP 源代码混在一起时会有麻烦。

6.1.3　PHP 程序注释方法

在 PHP 的程序中加入注释的方法很灵活。可以使用 C 语言、C++语言或 UNIX 的 Shell 语言的注释方式，也可以将它们混合使用。具体方法如下。

（1）“//”：这是从 C++语法中借鉴来的，该符号只能注释一行。

（2）“/*”和“*/”：这是 C 语言的注释符，符号之间的字符都为注释。

（3）“#”：这是 UNIX 的 Shell 语言风格的注释符，也只能注释一行。

下面的程序 6-2.php 中就用了 3 种不同风格的注释。

程序 6-2.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-2.php:php 的三种不同的注释方法-->
<head>
<title>6-2.php</title>
</head>
<body>
<?php
  /* 这个程序简单说明三种注释方式
   *当前在段注释方式中
   *本段文字不会得到显示
   */
```

```
    echo "使用/**/注释一段<p>";
     echo "使用//注释单行<p>";      //单行注释
    echo "使用 UNIX Shell 风格的#注释";     #"#"也只能注释一行
 ?>
</body>
</html>
```

其运行结果如图 6-2 所示。



图 6-2 程序 6-2.php 的运行结果

6.1.4 在 PHP 中引用文件

文件引用可以将常用的功能或代码写成一个文件，然后在需要的地方直接引用（调用）即可了。这样既可以简化程序代码又可以实现代码的复用。

引用文件的方法有两种：require 方法和 include 方法。这两种方法除了处理失败的方式不同之外完全一样。当产生错误时，使用 include()会产生一个警告，而使用 require()则导致一个致命错误。换句话说，如果想在遇到丢失文件时停止处理页面就用 require()，如果使用 include()遇到丢失文件时，脚本会继续运行。

require 的使用方法 require ("MyRequireFile.php")、require ('MyRequireFile.php')、require "MyRequireFile.php"、require 'MyRequireFile.php'都是正确的。

include 使用方法 include ("MyIncludeFile.php")、include ('MyRequireFile.php')、include "MyRequireFile.php"、include'MyRequireFile.php'都是正确的。

下面先建立一个名为 6-3.php 的文件，输入如下代码：

程序 6-3.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-3.php: php 文件的引用-->
<head>
<title>6-3.php </title>
</head>
<body>
  <?php
    echo"这是主文件"6-3.php"输出的! <br>";
    include("myFile.php");
```

```
        //引用同目录下名为"myFile.php"的 php 文件
        echo "<br>继续执行主文件"6-3.php"!";
      ?>
  </body>
  </html>
```

然后建立一个名为 myFile.php 的文件，其代码如下：

```
<!--文件 myFile.php：被"6-3.php"文件所引用的文件-->
<?php
 echo'这是从"myFile.php"文件中输出的. '."<br>";
 echo "2+6=".(2+6);
?>
```

其运行结果如图 6-3 所示。



图 6-3　程序 6-3.php 的运行结果

另外，还有 include_once()和 require_once()也可以用来引用文件，它们的行为与 include()和 require()语句类似，唯一的区别是如果该文件中的代码已经被包含了，则不会再次包含。可以进行测试：

（1）在 3-4.php 中的第 9 和第 10 行之间在加入一句“include("include.msp");”，执行程序会发现输出两句“这是从‘include.msp’文件中输出的！”。

（2）在 3-4.php 中的第 9 和第 10 行之间在加入一句“include_once ("include.msp");”，执行程序会发现只输出了一句“这是从‘include.msp’文件中输出的！”。

require()和 require_once()的区别也是一样的，也可以用上面的方法进行测试。

6.1.5　PHP 的命名规则

一个良好的命名规则能让代码变得更加清晰流畅，不仅方便阅读，而且易于维护。变量名应尽量简单明了，易于记忆，做到见名知意。除非是一次性的临时变量，尽量不用单个字符命名变量。PHP 变量名大小写敏感。类、实例、类常量应采用大小写混合的命名方法，即第一个单词的首字母小写，其后单词的首字母大写。语法上允许以_（下划线）和$（美元符号）开头，但实际应用中要尽量避免这种命名方法。

6.1.6　在 PHP 中输出 XHTML 代码

PHP 程序运行于服务器，运行得出的结果通过 PHP 的显示函数输出到浏览器页面中。

在 PHP 显示函数中使用 XHTML 代码可以使 PHP 输出更为美观的界面内容。例如，下面的代码：

```
<?php
echo '<h2 align="center">第六章</p>';
print "<br>";
echo "<p ><font size='3'> PHP 程序运行于服务器，运行得出的结果通过 PHP 的显示函数
输出到浏览器页面中。在 PHP 显示函数中使用 XHTML 代码可以使 PHP 输出更为美观的界面内容。
</font></p>";
?>
```

在使用 PHP 输出 XHTML 时，要特别注意单引号和双引号的嵌套。例如，如下代码是错误的：

```
echo '<p align='center'>单引号的嵌套</p>';
echo "<font size="5">双引号的嵌套</font>";
```

当定义字符串时，只有一种引号被视为定义符，即单引号或双引号。于是，如果一个字符串由双引号开始，那么只有双引号被分析器解析。这样，就可以在双引号串中包含任何其他字符，甚至单引号。当 PHP 遇到与串的开头相对应的引号时，便认为已经到了字符串尾部。因此在前面的例子里，成对的单引号里面使用双引号或成对的双引号中使用单引号都是正确的嵌套方式。

另一种方法是使用转义字符“\”将嵌套的引号转义。例如，上面的错误代码可以修正为：

```
echo '<p align=\'center\'>单引号的嵌套</p>';
echo "<font size=\"5\">双引号的嵌套</font>";
```

XHTML 代码还可以将嵌入到 PHP 标记之间来输出 XHTML，如程序 6-4.php。

程序 6-4.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-4.php:php 输出 XHTML-->
<head>
 <title>6-4.php</title>
 </head>
<body>
<?php
$num=5;
if($num)
{   for($i=1;$i<$num;$i++){
echo "编号为 ".$i;
?>
<font color="red" size="3">的同学作业已提交</font><br>
<?php
  }
}        //这里的代码与之前的代码是连接的
```

```
?>
</body>
</html>
```

程序运行结果如图 6-4 所示。

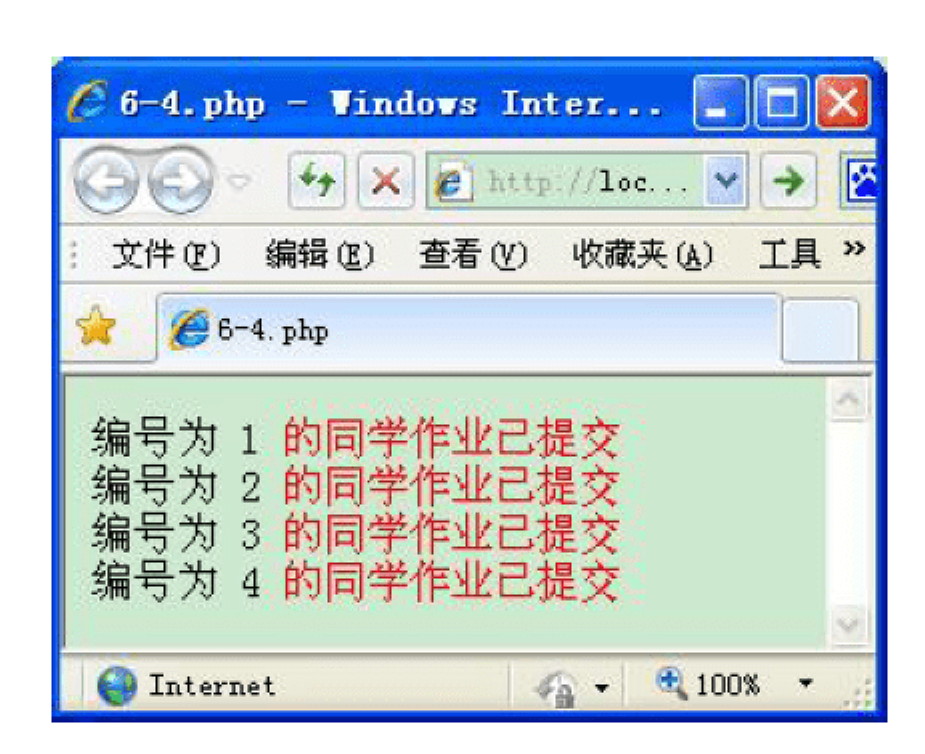


图 6-4 程序 6-4 运行结果

6.1.7 在 PHP 中使用 JavaScript

PHP 代码中嵌入 JavaScript 能够与客户端建立良好的用户交互界面，减轻服务器端的负担，强化 PHP 的功能，其应用十分广泛。PHP 在服务器端的设置，可以通过客户端的 JavaScript 表现出来，提升用户体验。在使用时应注意 PHP 生成 JavaScript 脚本的语法，不要与 JavaScript 语法混合。

在 PHP 中生成 JavaScript 脚本的方法与输出 XHTML 的方法一样，也是使用显示函数，如程序 6-5.php。

程序 6-5.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-5.php:php 输出 JavaScript-->
<head>
  <title>6-5.php</title>
</head>
<body>
<?php
$str="PHP 中的字符串";
echo "<script>";                //嵌入 JS
echo "alert('我是 JavaScript! ');";
echo "alert('".$str." ');";     //JS 中使用 PHP 的变量
echo "</script>";
?>
</body>
</html>
```

程序运行结果如图 6-5 所示。

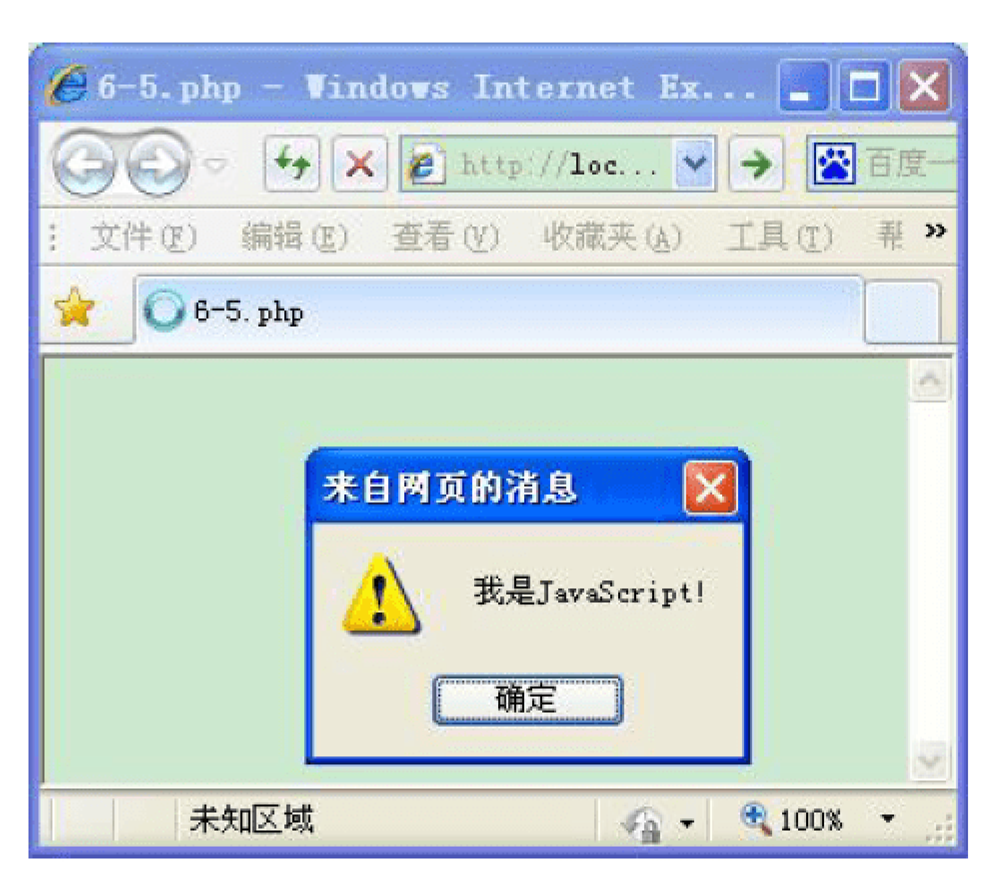


图 6-5　程序 6-5 运行结果

6.2　PHP 的数据类型

6.2.1　数据类型

计算机程序处理的是数据，每个数据都有其类型。PHP 中的数据类型可分为标量数据类型、复合类型和特殊类型，共包含 8 种不同的类型，如表 6-1 所示。

表 6-1　PHP 的数据类型

分　类	类　型	类 型 名 称
标量数据类型	boolean	布尔型
	integer	整型
	float	浮点型，也可以用 double
	string	字符串
复合类型	array	数组
	object	对象
特殊类型	resource	资源
	NULL	

下面分别介绍这 8 种数据类型。

1. 布尔型（boolean）

布尔型是最简单的类型，也被称为逻辑型，其值非真即假，主要用在条件表达式和逻辑表达式中，用以控制程序流程。要指定一个布尔值，使用关键字 TRUE 或 FALSE（两个都不区分大小写）。其他类型的数据均可以转换为布尔型，转换规则详见类型转换。

2. 整型（integer）

整型数的字长和平台有关。在 32 位的操作系统中，整型数据的有效范围是–2 147 483 648～2 147 483 647。

整型值可以用十进制、十六进制或八进制符号指定，前面可以加上可选的符号（–或者+）。要使用八进位整数可以在面加 0（零），要使用 16 进位整数可以在面加 0x。

3. 浮点型（double (floating point number)）

浮点数据类型用来存储实数，提供了比整型数据大得多的精度。在 32 位的操作系统中，它的有效范围是 1.7E–308～1.7E+308。例如：

```
$float1=666.66
$float2=6.6666e2    //表示 6.6666 乘以 10 的 2 次方，为指数形式的浮点数
```

值得注意的是，浮点型变量显示的十进制数的位数由 php.ini 文件中的 precision（精度）定义，预定值为 12，即浮点数最长占 14 字符。

4. 字符串（string）

无论是单个字符还是很长的字符串都是使用这个数据类型。定义字符串的方式有 3 种：

（1）单引号方式，如$str1=“I am studying PHP.”。

（2）双引号方式，如$str2=‘I am studying PHP.’。

（3）定界符方式，如：

```
$str3=<<<mark。
山东省日照市<br>
曲阜师大日照校区
mark;  //必须放在行首，不能缩进
echo  $str3;
```

当输出大段 XHTML 或 JavaScript 代码时，用（1）和（2）两种方法要使用大量的转义字符，容易出现语法错误。因此应当使用定界符方式输出。定界符的作用就是按照原样输出，且其中的任何特殊字符都不需要转义，变量也会替换成变量的值。

5. 数组（array）

数组类型可以是一维数组、二维数组或更多维数组，其中的元素可以为多种类型，可以是字符串、整型、浮点型、布尔型，甚至是数组或对象等。对数组的详细介绍见第 7 章。

6. 对象（object）

object 为对象类型，是类的具体化实例。

7. 资源（resource）

资源是一种特殊类型，其中保存了到外部资源的一个引用。资源是通过专门的函数建立和使用的。资源类型可保存打开的文件、数据库连接、图形画布区域等的特殊句柄。

8. NULL

NULL 类型只有一个值，就是大小写敏感的关键字 NULL，表示一个值为空。

在下列情况下一个变量的值被认为是 NULL。

（1）被赋值为 NULL。

（2）尚未被赋值。

（3）被 unset()（销毁指定的变量）。

6.2.2 变量类型转换

PHP 是一种弱类型的语言，变量的类型是赋给变量的数据的类型决定的，也就是说 PHP 在变量定义时不需要类型定义，变量的类型是根据使用该变量的上下文所决定的。如果把一个字符串值赋给一个变量，这个变量就成了一个字符串类型的变量；如果又把一个整型值赋给它，那它就成了一个整型变量，其值是赋给它的那个整数。

在 PHP 中是怎样处理变量不同类型间的相互转换的呢？PHP 提供了两种类型转换的方法：自动类型转换和强制类型转换。

PHP 的自动类型转换的一个例子是加号“+”。如果任何一个运算数是浮点数，则所有的运算数都被当成浮点数，结果也是浮点数。否则运算数会被解释为整数，结果也是整数。注意这并没有改变这些运算数本身的类型；改变的仅是这些运算数如何被求值。也就是说，自动类型转换并不能改变变量本身的数据类型，改变的仅仅是变量作为运算数时被求值的方式。

PHP 中的类型强制转换和 C 中的强制类型转换相似：在要转换的变量之前加上用括号括起来的目标类型。允许的强制转换如下。

（int）或（integer）：转换成整型。

（bool）或（boolean）：转换成布尔型。

（float）、（double）或（real）：转换成浮点型。

（string）：转换成字符串。

（array）：转换成数组。

（object）：转换成对象。

其使用方法如下：

```
(int) $变量名或 (integer) $变量名
```

为了便于理解，现举例程序 6-6.php 如下。

程序 6-6.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-6.php：变量类型转换-->
<head>
<title>6-6.php</title>
</head>
<body>
  <?php
   $var1 = "0";                  // $var1 是一个字符串
   echo $var1."<br>";
   $var2 =$var1 + 2;             // $var2 是一个整数
   echo $var2."<br>";
   $var3 = $var2 + 1.3;          // $var3 是一个浮点数
   echo $var3."<br>";
   $var4 = 5 + "10 PHP5.2";      //整数与字符串相加结构为正数
   echo $var4."<br>";
   $var5 = 5 + "PHP5.2";         //整数与字符串相加结构为正数
```

```
    echo $var5."<br>";
    $var6 = (bool)-2;          //-1 和其他非零值（不论正负）转换成布尔型，都
                                 被认为是 TRUE!
    echo $var6."<br>";
    $var7 = 10/3;
    echo $var7."<br>";
    $var8 = (int)$var7;      //强制转换为整数
    echo $var8."<br>";
    $var9=1.3e5;
    $var10 = (float)$var9;   //强制转换为浮点数
    echo $var10."<br>";
    $var11 = (string)$var3;
echo "\$var11 的类型为：".gettype($var11)."<br>";
                               //gettype()为获取变量类型的函数
  ?>
</body>
</html>
```

程序 6-6.php 运行结果如图 6-6 所示。

程序 6-6.php 中仅举例说明了部分类型转换，转换为布尔值时需要注意以下几点。当转换为 boolean 时，以下值被认为是 FALSE。

（1）布尔值 FALSE。

（2）整型值 0（零）。

（3）浮点型值 0.0（零）。

（4）空白字符串和字符串"0"。

（5）没有成员变量的数组。

（6）没有单元的对象（仅适用于 PHP 4）。

（7）特殊类型 NULL（包括尚未设定的变量）。

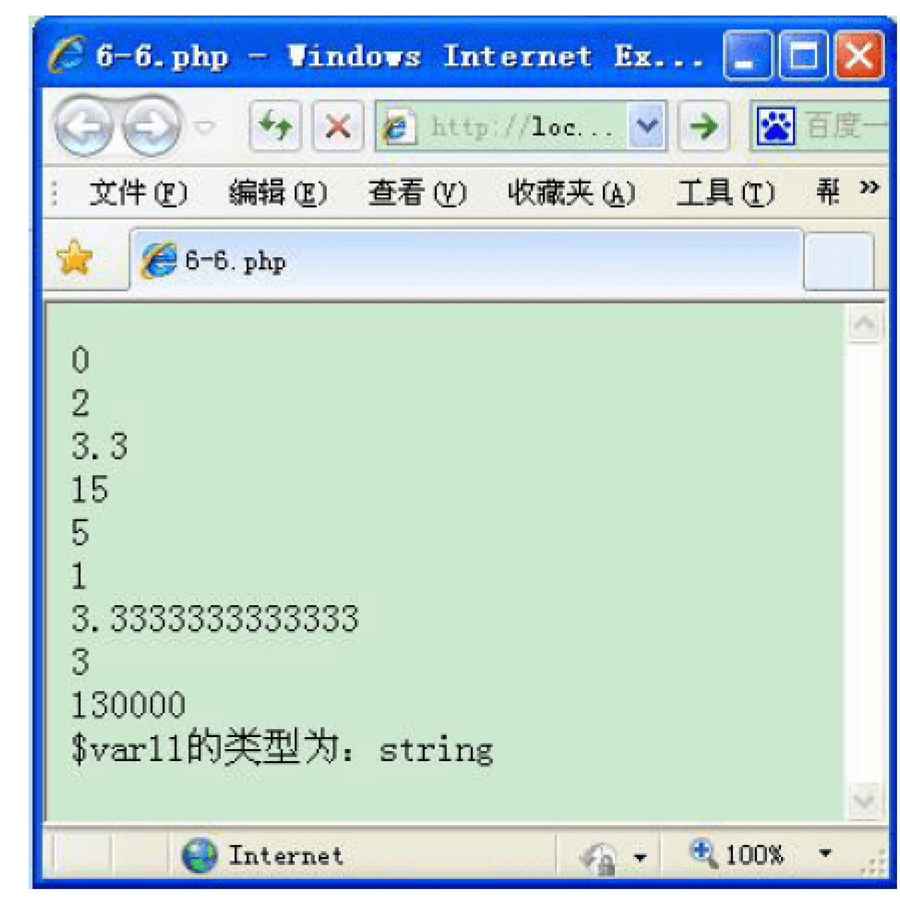


图 6-6　程序 6-6.php 的运行结果

所有其他值都被认为是 TRUE（包括任何资源）。值得注意的是，−1 和其他非零值（不论正负）一样，被认为是 TRUE，在求表达式的值和条件判断时一定要注意。

6.3　PHP 中的变量

变量是指在程序执行过程中其值可以变化的量，系统为程序中的每一个变量分配一个存储单元。变量通过变量名来标识，变量名就是存储变量数据的存储单元名。PHP 中的变量有两种：自定义变量和预定义变量。下面先学习自定义变量。

6.3.1　变量定义与赋值

在 PHP 中一个有效的变量名由字母或者下划线开头，后面跟上任意数量的字母、数字或下划线。PHP 的变量属于松散的数据类型，具体使用时应注意以下几点。

- 变量名要以“$”开头，且区分大小写；
- 变量不必要预先定义或声明；

- 变量在使用时编译器可动态进行类型指定和转换；
- 变量如果未赋值而直接使用，变量值将被视为空。

以下变量命名都是正确的：

```
$num=12;
$my_name="root";
$_class="09-2";
$_1245=67.9;
```

以下变量命名则是错误的：

```
$23ab=112;   //变量名不能以数字开头
$班级="09-2";//变量名不能包含汉字
```

PHP 中给变量赋值有两种方式：传值赋值和引用赋值。

传值赋值方式使用“=”直接将一个变量（或表达式）的值赋给变量。这种方式，是通过在存储区域复制一个变量的副本来实现的，等号两端的变量值互不影响。

引用赋值同样也用“=”将一个变量的值赋给另一个变量，但还要在等号右端的变量前面加上一个“&”符号。这种方式不是将一个变量的副本赋给变量，而是将一个变量的地址赋给了另一个变量，改变任何一个变量的值都将引起另一个变量相应的变化。程序 6-7 演示了多种变量的使用。

程序 6-7.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--程序 6-7.php：PHP 变量的使用=>布尔型、整型、浮点型、字符串-->
<head>
<title>6-7.php</title>
</head>
<body>
  <?php
    $string1="输出字符串变量类型的内容！";
    echo $string1;          //输出字符串变量"$string"的内容
    echo "<br>";            //输出换行
    $string2="输出特殊字符：";
    echo $string2."\\";
    echo "\$";
    echo '\'';
    echo "\"";
    echo "\x52";
    echo "<br>";
    $int1=01234;            //八进制整数
    $int2=0x1234;
    echo "输出整型变量的值：";
    echo $int1;             //输出 668
    echo "\t";              //输出一个制表位
    echo $int2;             //输出 4660
    echo "<br>";
    $float1=6.6666e2;
    echo "输出浮点型变量的值：";
```

```
    echo $float1;            //输出 666.66
    echo "<br>";
    echo "输出布尔型变量的值: ";
    echo (boolean)($int1);   //输出转换后的布尔变量 1
   ?>
</body>
</html>
```

程序 6-7 涉及字符串、整型、浮点型、布尔型等类型变量的使用，运行结果如图6-7所示。

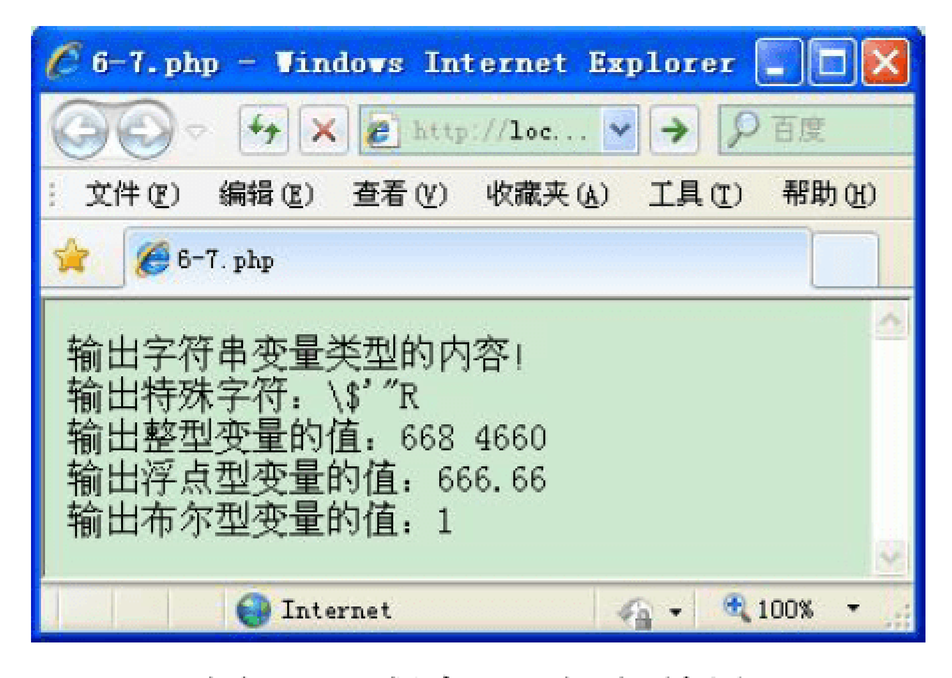


图 6-7　程序 6-7 运行结果

6.3.2　变量的检测

1. 检测变量是否设置

为了保证代码在各种设置下都能够安全运行，最好在使用变量前判断是否定义了该变量。PHP 中有两个能够完成此功能的函数：isset()和 empty()。

（1）isset()函数。函数声明如下：

```
bool isset(mixed var[,mixed var[,…]])
```

如果 var 存在，则返回 TURE，否则返回 FALSE。若用 isset()测试一个被设置成 NULL 的变量，返回值为 FALSE。

（2）empty()函数。函数声明如下：

```
bool empty(mixed var)
```

如果参数 var 是非空或非零的值，函数返回 FALSE。即“ ”、0、“0”、NULL、FALSE、array()以及没有任何属性的对象都将被认为是空的，如果 var 为空，则返回 TRUE。

下面的程序 6-8.php 演示了这两个函数的使用。

程序 6-8.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-8.php: 变量检测-->
<html>
<head>
  <title>6-8.php</title>
```

```
</head>
<body>
<?php
$age=0;
$weight=0.2;
$name="";
if(isset($age))
 echo '1 $age 已经定义了<br>';
 else
   echo '1 $age 还没有定义<br>';
if(empty($age))
  echo '2 $age 为空<br>';
else
  echo '2 $age 不空<br>';
if(isset($weight))
echo '3 $weight 已经定义了<br>';
 else
   echo '3 $weight 还没有定义<br>';
if(empty($weight))
  echo '4  $weight 为空<br>';
else
  echo '4  $weight 不空<br>';
if(isset($name))
echo '5 $name 已经定义了<br>';
 else
   echo '5 $name 还没有定义<br>';
if(empty($name))
  echo '6 $name 为空<br>';
else
  echo '6 $name 不空<br>';
 if(isset($brother))
echo '7 $brother 已经定义了<br>';
 else
   echo '7 $brother 还没有定义<br>';
if(empty($brother))
  echo '8 $brother 为空<br>';
else
  echo '8 $brother 不空<br>';
?>
</body>
</html>
```

程序运行结果表明，对于没有定义的变量，两个函数都能正确判断，但对值 0 和空字符串，两个函数判断的结果并不一致。程序运行结果如图 6-8 所示。

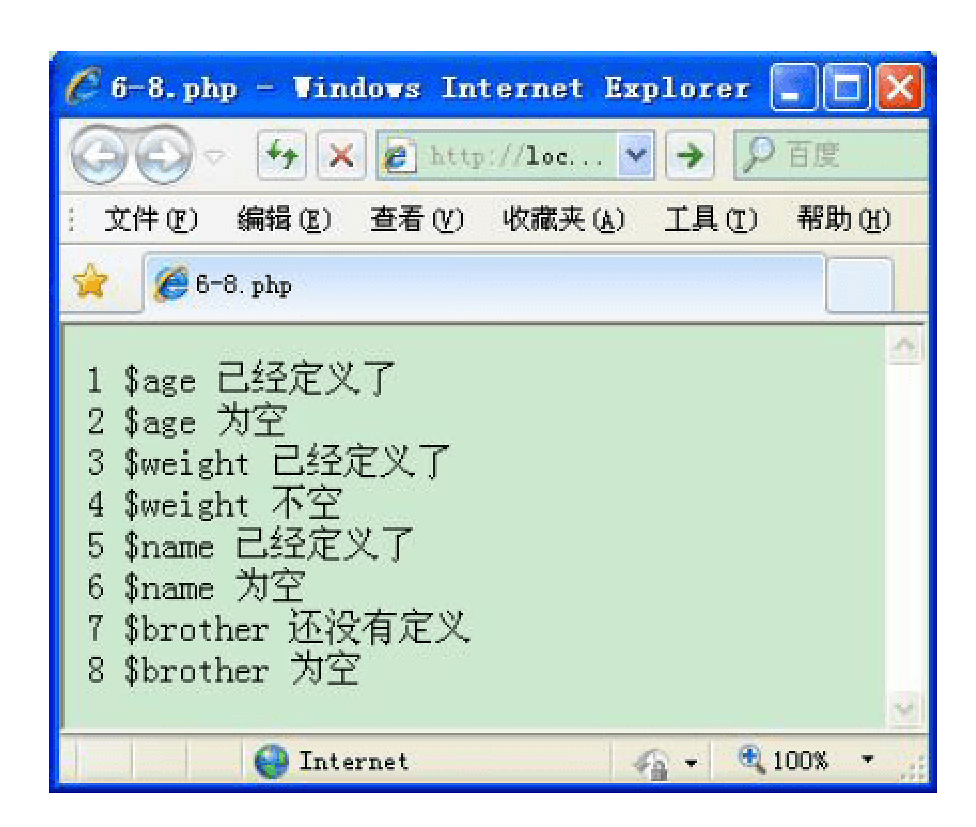


图 6-8 程序 6-8 运行结果

2. 检测变量类型

PHP 变量是弱类型的，实际类型取决于所赋值的类型。由于程序运行中变量中存储的值

可能经常变化，为保证程序运行的安全性，需要检测变量的类型。检测变量的类型有两类方法：var_dump()函数和 is_xxx()函数。

（1）var_dump()函数。函数声明如下：

```
void var_dump(minxed expression[,mixed expression[,…]])
```

其中，参数 expression 为变量名或表达式。

（2）is_xxx()函数。常用的函数有：

is_bool() 判断是否为布尔型；
is_float()判断是否为浮点型；
is_int()判断是否为整型；
is_numeric()判断是否为数值型；
is_string()判断是否为字符串；
is_array()判断是否为数组；
is_object()判断是否为对象。

程序 6-9.php 演示了检测变量类型函数的用法。

程序 6-9.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-9.php：变量类型判断-->
<head>
  <title>6-9.php</title>
</head>
<body>
<?php
$float=2.14;
$boor=FALSE;
$int=123456789;
$str="hello";
//var_dump()函数判断变量类型
echo '<br>$float 的类型是：';
var_dump($float);
echo '<br>$boor 的类型是：';
var_dump($boor);
echo '<br>$int 的类型是：';
var_dump($int);
echo '<br>$str 的类型是：';
var_dump($str);
// is_xxx 函数判断变量类型
if(is_float($float))
echo '<br>$float 是浮点型';
if(is_int($int))
echo '<br>$int 是整型';
if(is_string($str))
echo '<br>$str 是字符串型';
if(is_bool($boor))
```

```
echo '<br>$boor 是布尔型';
?>
</body>
</html>
```

程序运行结果如图 6-9 所示。

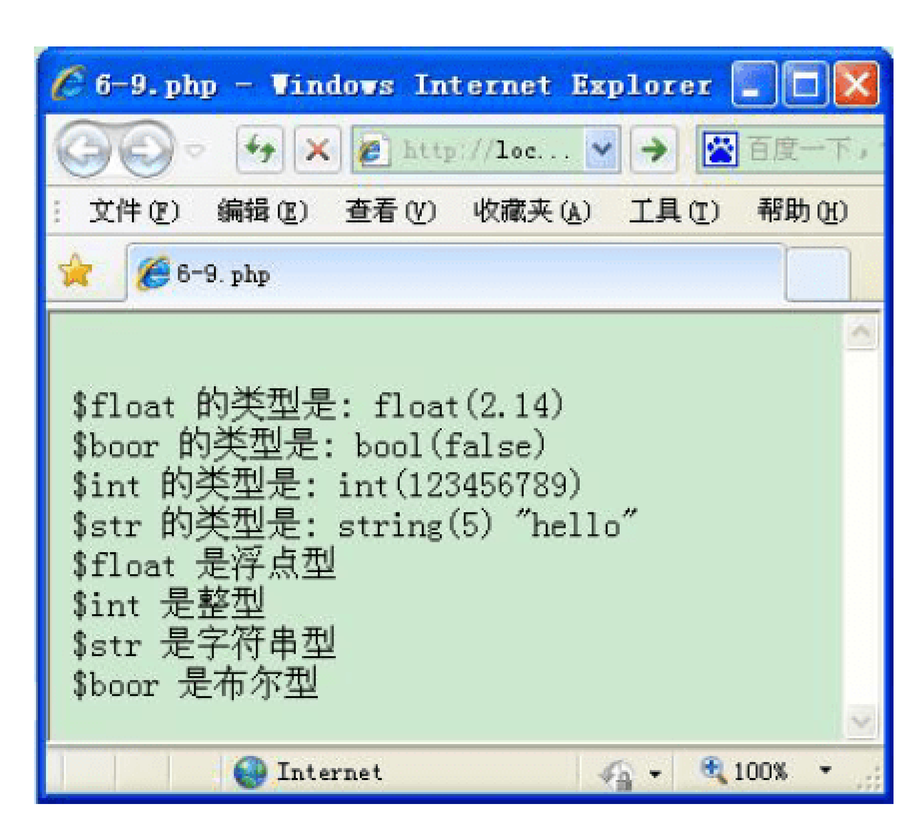


图 6-9 程序 6-9 运行结果

6.3.3 变量的作用域

变量的作用域是指变量的有效范围，即变量在哪些范围内能被使用，在哪些范围内不能被使用。根据作用域的不同，PHP 中的变量可分为局部变量和全局变量。

1. 局部变量

局部变量只是局部有效，它的作用域分为两种：

（1）在当前文件主程序中定义的变量，其作用域限于当前文件的主程序，不能在其他文件或当前文件的局部函数中起作用。如果通过 include()或 require()包含了其他文件，变量的作用域还可以扩展到这些文件中。

（2）在局部函数或方法中定义的变量仅限于局部函数或方法，当前文件中主程序、其他函数、其他文件中无法引用。

2. 全局变量

全局变量可以在程序的任何地方被访问。通过使用全局变量，能够实现在函数内部引用函数外部的参数，或在函数外部引用函数内部的参数。声明一个全局变量，只要在变量名前加上 global 关键字即可。

程序 6-10.php 为全局变量和局部变量使用示例。

程序 6-10.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-10.php: 局部变量和全局变量-->
<head>
   <title>6-10.php</title>
</head>
```

```
<body>
  <?php
   $var1=6;  //定义一个局部变量
   echo '定义局部变量\$var='.$var1;
   include ("a.php");
   function myfun1(){
       echo "<br>在 myfun1()函数中输出局部变量";
       echo "\$var1=".$var1;
            }
   myfun1();
   function myfun2(){
      echo "<br>在 myfun2()函数中输出局部变量";
      echo "\$var1=".$GLOBALS["var1"];
         }
   myfun2();
     global $var2;
     $var2=8;
     function myfun3(){
     global $var3;
     $var3=9;
     echo "<br>在 myfun3()函数中输出全局变量";
     echo "\$var2=".$var2;
     echo " \$var3=".$var3;
          }
     myfun3();
     echo "<br>在函数外输出全局变量";
     echo "\$var2=".$var2;
     echo " \$var3=".$var3;
    ?>
</body>
</html>
```

被包含的 a.php 代码如下：

```
<!--程序 a.php：程序 6-10.php 所包含的文件-->
<?php
    echo "<br>在 a.php 中输出\$var 的值";
    echo "\$var1=".$var1;
    echo "<br>在 a.php 中的函数中输出\$var 的值";
    function myfun4(){
        echo "\$var1=".$var1;
    }
?>
```

程序 6-10.php 的运行结果如图 6-10 所示。

注意：为了防止程序中变量的混乱，建议尽量不要使用全局变量，至少应尽可能少地使用全局变量。

3. 静态变量

静态变量是局部变量的特殊情况，不同之处在于静态变量能够在所属函数执行完之后仍保

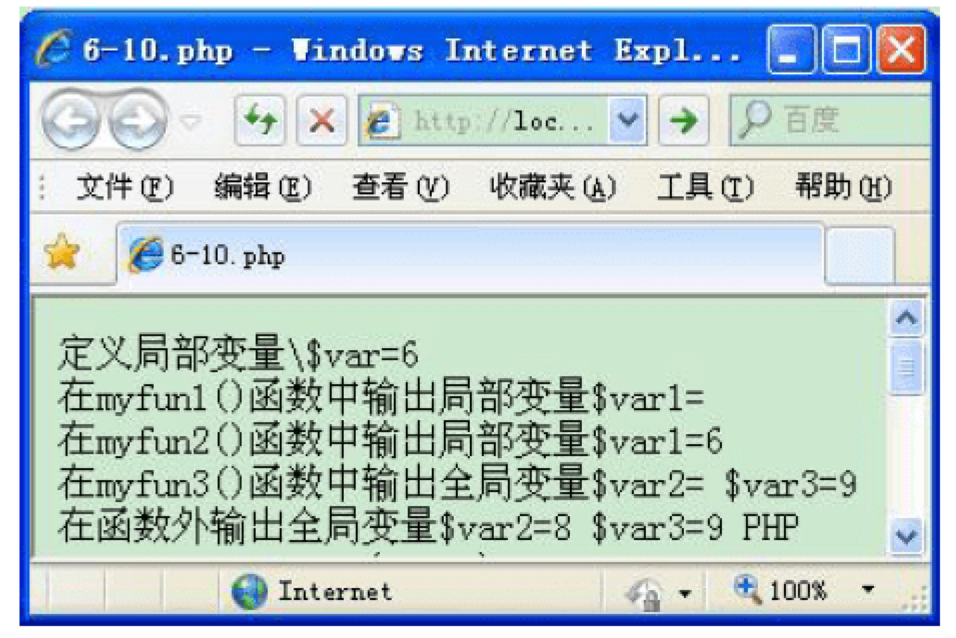


图 6-10　程序 6-10.php 的运行结果

留变量的值。当再次回到其作用域时，可以继续使用原来的值。而一般变量在函数调用结束后，其值将被清除，所占内存空间也被释放。

静态变量只能用于函数范围内，静态变量的定义方法为在变量名前加 static 关键字。程序 6-11.php 为静态变量示例。

程序 6-11.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-11.php：静态变量-->
<head>
  <title>6-11.php</title>
</head>
<body>
 <?php
  function fun1()
  {
   $int1=0;           //定义普通变量
   echo $int1." ";
   $int1++;
   }
  fun1();             //输出 0
  fun1();             //输出还是 0
  echo "<br>*************<br>";
  function fun2()
   {
    static $int2=0; //定义静态变量
    echo $int2." ";
    $int2++;
     }
  fun2();             //输出 0
  fun2();             //输出 1
  fun2();
 ?>
</body>
</html>
```

程序 6-11.php 的运行结果如图 6-11 所示。



图 6-11 程序 6-11.php 的运行结果

通过程序 6-11.php 不难看出静态变量与普通变量的区别，在函数 fun2 中静态变量$int2 只被初始化了一次，在该函数退出时$int2 的值不会丢失，所以反复调用函数 fun2 时，$int2 的值会累加。

6.3.4 PHP 的预定义变量

PHP 提供了一套附加的内置数组（也称为预定义数组或预定义变量），包含来自 Web 服务器（如果可用）、运行环境和用户输入的数据。这些数组非常特别，在全局范围内自动生效。因此通常被称为自动全局变量（autoglobals）或超全局变量（superglobals）。PHP 中

没有用户自定义超全局变量的机制。超全局变量主要有以下几个。

1. $GLOBALS（Global 变量）

它包含引用指向每个当前脚本的全局范围内有效的变量，即为由所有已定义全局变量组成的数组。该数组的索引为全局变量的变量名。

2. $_SERVER（服务器变量）

$_SERVER 是一个包含诸如头信息、路径和脚本位置的数组。数组的实体由 Web 服务器创建。不能保证所有的服务器都能产生所有的信息，服务器可能忽略了一些信息，这与服务器的设定或直接与当前脚本的执行环境相关联。

3. $_GET（HTTP GET 变量）

$_GET 通过 HTTP GET 方法传递的变量组成的数组。

4. $_POST（HTTP POST 变量）

$_POST 通过 HTTP POST 方法传递的变量组成的数组。

5. $_COOKIE（HTTP Cookies）

$_COOKIE 通过 HTTP Cookies 传递的变量组成的数组。

6. $_FILES（HTTP 文件上传变量）

$_FILES 通过 HTTP POST 方法传递的已上传文件项目组成的数组。

7. $_ENV（环境变量）

$_ENV 从环境变量通过执行转变过来的 PHP 全局变量。它们中的许多都是由 PHP 所运行的系统决定。

8. $_REQUEST（Request 变量）

$_REQUEST 经由 GET、POST 和 COOKIE 机制提交至脚本的变量，关联数组包含$_GET、$_POST 和$_COOKIE 中的全部内容。该数组并不值得信任，建议尽量少用，甚至不用。所有包含在该数组中的变量的存在与否以及变量的顺序均按照 php.ini 中的 variables_order 配置指示来定义。

9. $_SESSION（Session 变量）

$_SESSION 包含当前脚本中 Session 变量的数组。

10. $php_errormsg（前一个错误消息）

$php_errormsg 是包含 PHP 产生的上一错误消息内容的变量。该变量在发生错误并且将 track_errors 选项打开（默认为关闭）后才有效。

其中，$_GET、$_POST、$_SESSION、$_COOKIE 等预定义变量常用于 PHP 文件间的通信，以构建动态网页，这几个预定义变量的使用将在 6.7 节中详细介绍。下面的程序 6-12.php 演示了用$_SERVER 获取服务器信息的方法。

程序 6-12.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-12.php：服务器信息的获取-->
<head>
  <title>6-12.php</title>
</head>
<body>
```

```
   <?php
      echo "1、".$_SERVER["PHP_SELF"]."<br>";
      echo "2、".$_SERVER["argv"]."<br>";
      echo "3、".$_SERVER["argc"]."<br>";
      echo "4、".$_SERVER["SERVER_NAME"]."<br>";
      echo "5、".$_SERVER["SERVER_SOFTWARE"]."<br>";
      echo "6、".$_SERVER["SERVER_PROTOCOL"]."<br>";
      echo "7、".$_SERVER["REQUEST_METHOD"]."<br>";
      echo "8、".$_SERVER["REQUEST_TIME"]."<br>";
      echo "9、".$_SERVER["QUERY_STRING"]."<br>";
      echo "10、".$_SERVER["DOCUMENT_ROOT"]."<br>";
      echo "11、".$_SERVER["HTTP_ACCEPT"]."<br>";
      echo "12、".$_SERVER["HTTP_ACCEPT_ENCODING"]."<br>";
      echo "13、".$_SERVER["HTTP_ACCEPT_LANGUAGE"]."<br>";
      echo "14、".$_SERVER["HTTP_CONNECTION"]."<br>";
      echo "15、".$_SERVER["HTTP_HOST"]."<br>";
      echo "16、".$_SERVER["HTTP_USER_AGENT"]."<br>";
      echo "17、".$_SERVER["HTTPS"]."<br>";
      echo "18、".$_SERVER["REMOTE_ADDR"]."<br>";
      echo "19、".$_SERVER["REMOTE_HOST"]."<br>";
      echo "20、".$_SERVER["SCRIPT_FILENAME"]."<br>";
      echo "21、".$_SERVER["SERVER_PORT"]."<br>";
      echo "22、".$_SERVER["PATH_TRANSLATED"]."<br>";
      echo "23、".$_SERVER["SCRIPT_NAME"]."<br>";
      echo "24、".$_SERVER["REQUEST_URI"]."<br>";
    ?>
</body>
</html>
```

程序 6-12.php 运行结果如图 6-12 所示。



图 6-12 程序 6-12 运行结果

文件上传的功能是经常使用的，这要用到$_FILE 数组。有了文件上传的功能，不仅可以为网站动态添加附件，以实现网页的文字编辑功能，而且还可以实现网站中相关图片、Flash 动画等的动态更新等。下面就通过程序 6-13.php 来了解$_FILE 数组的使用方法和文件上传的基本原理。

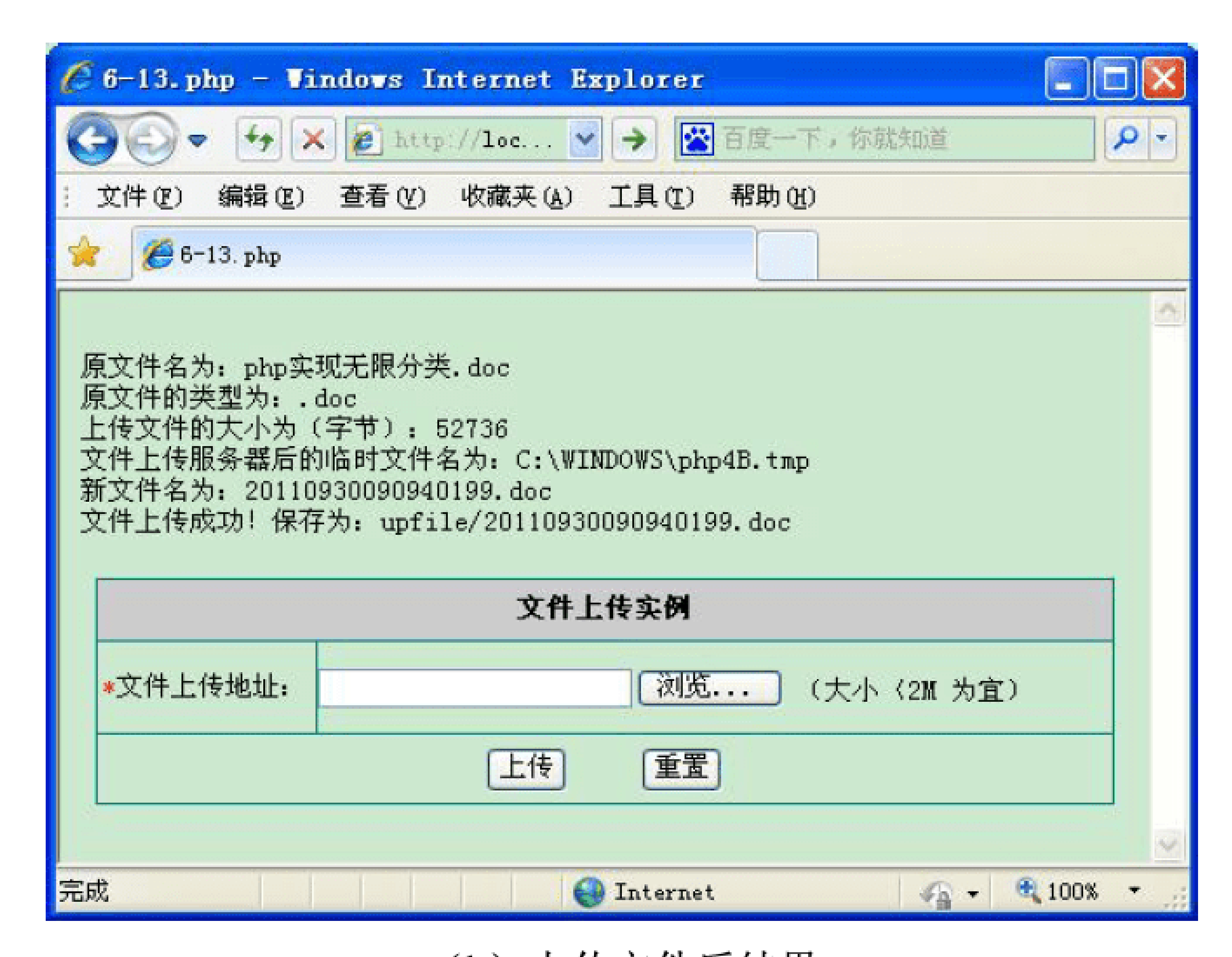
程序 6-13.php

```
<!--文件 6-13.php：文件上传实例-->
<!--为能运行，请在本文件所在目录新建一名为"upfile"的文件夹，权限设置可写-->
<?php
if ($_POST[add]=="上传"){
       //根据现在的时间产生一个随机数
       $rand1=rand(0,9);
       $rand2=rand(0,9);
       $rand3=rand(0,9);
       $filename=date("Ymdhms").$rand1.$rand2.$rand3;
       if(empty($_FILES['file_name']['name'])){
       //$_FILES['file_name']['name']为获取客户端机器文件的原名称
           echo "文件名不能为空";
           exit;
           }
       $oldfilename=$_FILES['file_name']['name'];
       echo "<br>原文件名为：".$oldfilename;
$filetype=substr($oldfilename,strrpos($oldfilename,"."),strlen($oldfilename)-strrpos($oldfilename,"."));
 echo "<br>原文件的类型为：".$filetype;
 if(($filetype!='.doc')&&($filetype!='.xls')&&($filetype!='.DOC')&&($filetype!='.XLS')){
    echo "<script>alert('文件类型或地址错误');</script>";
    echo "<script>location.href='6-13.php';</script>";
    exit;
     }
    echo "<br>上传文件的大小为（字节）：".$_FILES['file_name']['size'];
     //$_FILES['file_name']['size']为获取客户端机器文件的大小，单位为 B
    if ($_FILES['file_name']['size']>1000000) {
    echo "<script>alert('文件太大，不能上传');</script>";
    echo "<script>location.href='6-13.php';</script>";
     exit;
     }
   echo "<br>文件上传服务器后的临时文件名为：".$_FILES['file_name']['tmp_name'];
    //取得保存文件的临时文件名（含路径）
   $filename=$filename.$filetype;
   echo "<br>新文件名为：".$filename;
   $savedir="upfile/".$filename;
   if(move_uploaded_file($_FILES['file_name']['tmp_name'],$savedir)){
   $file_name=basename($savedir);       //取得保存文件的文件名（不含路径）
      echo "<br>文件上传成功！保存为：".$savedir;
      }else{
       echo "<script language=javascript>";
       echo "alert('错误，无法将附件写入服务器！\n 本次发布失败！');";
```

```
        echo "location.href='5-6.php?';";
        echo "</script>";
         exit;
         }
    }
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Language" content="zh-cn">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>==文件上传实例==</title>
<style>
body{font-size:10pt};
td{font-size:10pt};
.style1 {color: #FF0000}
.style2 {
    color: #000000;
    font-weight: bold;
}
</style>
</head>
<body>
<div align="center">
</div>
  <form  enctype="multipart/form-data" action=" " method="post">
  <!--此处一定要有enctype="multipart/form-data"//-->
  <table width="486" height="103" border="1" align="center" cellpadding=
"0" cellspacing="0" bordercolor="#008080" id="AutoNumber1" style="border-
collapse: collapse">
<tr bgcolor="#CCCCCC">
 <td  height="30"  colspan="2"  align="right"><div  align="center"
class="style2">文件上传实例</div>
  </td></tr>
 <tr>
 <td width="103" height="45" align="right"><div align="center"><span
class="style1"> *</span>文件上传地址：</div></td>
  <td width="377" height="45"><input type="file" name="file_name">
  （大小〈2M 为宜）</td>
  </tr>
  <tr>
  <td height="33" align="right" colspan="2">
   <p align="center"><input type="submit" value="上传" name="add">
        <input  type="reset"  value=" 重 置 "  name=
"B2"></td>
  </tr>
 </table>
 </form>
 </body>
 </html>
```

程序 6-13.php 运行结果如图 6-13 所示。

（a）上传文件前结果



（b）上传文件后结果

图 6-13　程序 6-13 运行结果

6.3.5　可变变量

可变变量是一种独特的变量，允许动态地改变一个变量的名称。其工作原理是，该变量的名称由另外一个变量的值确定。声明一个可变变量的方法是在该变量名前加两个“$”符号。通过程序 6-14.php 就很容易理解它的使用方法。

程序 6-14.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-14.php：变量的变量-->
<head>
  <title>6-14.php</title>
</head>
```

```
08  <body>
09    <?php
10      $var = 'hello';
11      $$var = ' world! ';      //定义可变变量
12      echo $var.$$var."<br>"; //输出可变变量
13      echo $hello;            //输出可变变量
14    ?>
15  </body>
16  </html>
```

程序 6-14.php 的运行结果如图 6-14 所示。

程序中第 11 行，语句“$$var='world!';”定义了一个可变变量，相当于“$hello='world!';”，因有“$var='hello';”，将“$var”替换为 hello，即用一个变量的值作为新变量的变量名。



图 6-14　程序 6-14.php 的运行结果

6.4　PHP 中的常量

常量用于存储程序中不经常改变的数据信息。PHP 的常量有两种：一种是系统预定义常量；另一种是自定义常量。

6.4.1　预定义常量

PHP 为运行的脚本提供了大量的预定义常量，用于获取系统配置信息、网络请求等相关信息。

下面列举了一些常用的系统预定义常量：

1. __FILE__

本默认常量是文件的完整路径和文件名。如果用在包含文件中，则返回包含文件名。

2. __LINE__

本默认常量是文件中的当前行号。如果用在包含文件中，则返回在包含文件中的当前行号。

3. PHP_VERSION

本内建常量为 PHP 程序的版本，如' 5.2.5'。

4. PHP_OS

本内建常量指执行 PHP 解析器的操作系统名称，如'Linux'。

5. TRUE

本常量就是真值（true）。

6. FALSE

本常量就是假值（false）。

7. E_ERROR

本常量指到最近的错误处。

8. E_WARNING

本常量指到最近的警告处。

9. E_PARSE

本常量为解析语法有潜在问题处。

10. E_NOTICE

本常量为发生不正常但不一定是错误处，如存取一个不存在的变量。

11. NULL

一个 NULL 值。

这些以 E_开头的常量，可以参考 error_reporting 函数，其中有更多的相关说明。下面的程序 6-15.php 就是利用系统预定义常量输出一些系统参数。

程序 6-15.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-15.php: PHP 预定义常量-->
<head>
  <title>6-15.php</title>
</head>
<body>
 <?php
  echo(__LINE__);     //输出 8
  echo "<p>";
  echo(__FILE__);
  echo "<p>";
  echo(__LINE__);     //输出 12
  echo "<p>";
  echo PHP_VERSION;
  echo "<p>";
  echo PHP_OS;
 ?>
</body>
</html>
```

程序 6-15.php 运行结果如图 6-15 所示。

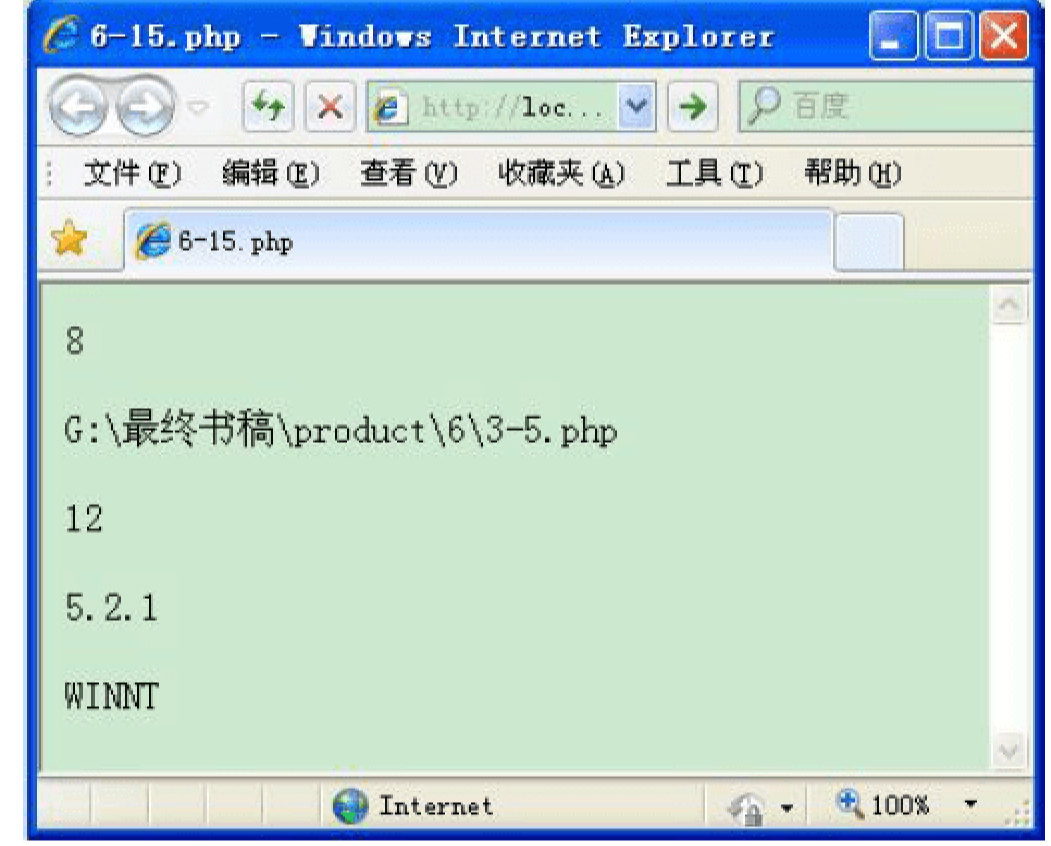


图 6-15　程序 6-15.php 的运行结果

6.4.2　自定义常量

编写程序时仅使用以上的系统预定义常量是不够的，函数 define 可以让用户自行定义所需要的常量，其定义的语法为：

```
bool define (string name, mixed value[, bool case_insentive])
```

其中，第一个参数为常量名；第二个参数为常量值；第三个参数指定参数名是否大小写敏感，设定为 TRUE 时表示不敏感。

自定义常量在定义和使用时应注意以下几点。

（1）常量只能用 define 函数定义，而不能通过赋值语句。

（2）常量前面没有美元符号（$）。

（3）常量可以不用理会变量范围的规则而在任何地方定义和访问。

（4）常量一旦定义之后，在其他地方不得再次定义同名的常量或变量。

（5）常量的值只能是标量（boolean、integer、float 和 string）。

（6）常量是区分大小写的，建议尽量全部使用大写字符，便于阅读和识别。

（7）默认情况下，常量是全局的，即在当前教本的任何地方都可使用。

（8）要判断一个常量是否已经定义，使用 defined 函数。

自定义常量的用法详见程序 6-16.php。

程序 6-16.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-16.php: PHP 自定义常量-->
<head>
 <title>6-16.php</title>
</head>
<body>
<?php
define("PI",3.14159,TRUE);
echo PI."<br>";
echo pi."<br>";
define("MAXSIZE",100);
echo MAXSIZE."<br>";
echo(defined("PI"));
echo "<br>";
?>
</body>
</html>
```

其运行结果如图 6-16 所示。

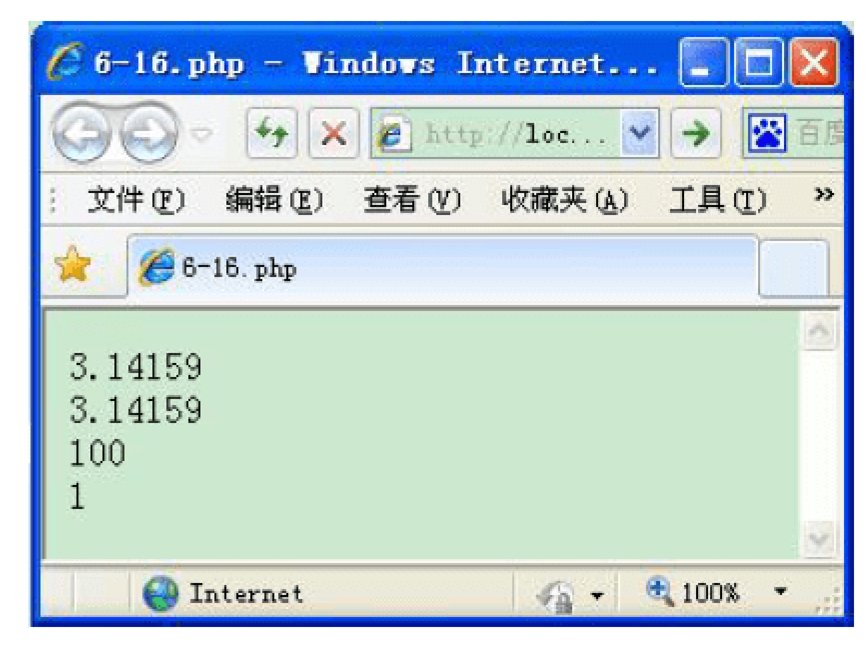


图 6-16 程序 6-16.php 的运行结果

6.5 运算符和表达式

6.5.1 PHP 的运算符

PHP 的运算符大部分是从 C 语言中借用而来的，分为以下几类。

算术运算符：+、−、*、/、%、++、−−。

字符串运算符：• 。

赋值运算符：=、+=、−=、*=、/=、%=、.=。

位运算符：& 、| 、^ 、<< 、>> 、~。

逻辑运算符：&& (and)、|| (or)、xor (xor)、! (not)。

比较运算符：<、>、<=、>=、==、===、!=。

其他运算符：$、&、@、−>、=>、?:。

下面就详细讲解各类运算符。

1. 算术运算符

算术运算符是用来处理四则运算的符号，是最简单也最常用的符号。尤其是数字的处理，几乎都会使用到算术运算符号，其具体意义如表 6-2 所示。

表 6-2　PHP 的运算符

符　　号	意　　义	符　　号	意　　义
+	加法运算	%	取余数
−	减法运算	++	自加
*	乘法运算	−−	自减
/	除法运算		

程序 6-17.php 为算术运算符的使用示例。

程序 6-17.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-17.php：算术运算符的应用-->
<head>
 <title>6-17.php</title>
</head>
<body>
<?php
$a=6;
 $b=3;
 echo $a."+".$b."=";
 echo $a+$b."<br>";
 echo $a."-".$b."=";
 echo $a-$b."<br>";
 echo $a."*".$b."=";
 echo $a*$b."<br>";
 echo $a."/".$b."=";
 echo $a/$b."<br>";
 echo $a."%".$b."=";
 echo $a%$b."<br>";
 echo $a."++=";
 echo $a++."<br>";
 $a=10;
 echo "++".$a."=";
 echo ++$a."<br>";
 $a=10;
 echo $a."--=";
 echo $a--."<br>";
 $a=10;
 echo "--".$a."=";
 echo --$a."<br>";
 $c="b";
 echo "\"b\"++=";
```

```
 $c++;
 echo "\"".$c."\"<br>";
?>
</body>
</html>
```

程序 6-17.php 的运行结果如图 6-17 所示。

2. 字符串运算符

字符串运算符号只有一个，就是英文的句号“.”。它可以将字符串连接起来，变成合并的新字符串，也可以将字符串与数字连接，这时类型会自动转换，具体用法如程序 6-18.php 所示。

程序 6-18.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-18.php：字符串运算符的应用-->
 <head>
    <title>6-18.php</title>
</head>
<dody>
 <?php
   $name="小叮当";
   $class="11 级 2 班";
   echo "我的名字是".$name."<br>";
   echo "我的班级是".$class;
  ?>
</body>
</html>
```

程序 6-18.php 的运行结果如图 6-18 所示。



图 6-17 程序 6-17.php 的运行结果

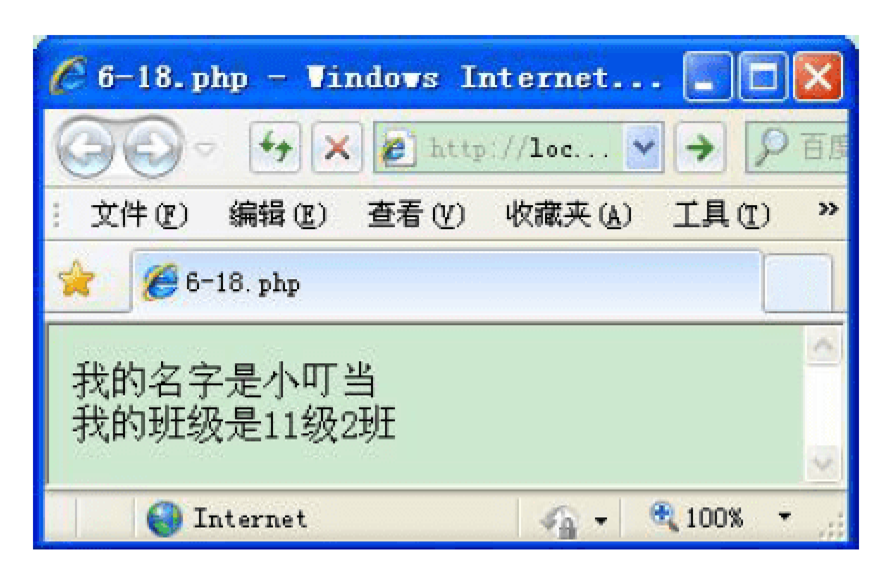


图 6-18 程序 6-18.php 的运行结果

3. 赋值运算符

赋值运算符的具体意义如表 6-3 所示。

表 6-3　PHP 的赋值运算符

符　号	意　义
=	将右边的值赋给左边的变量
+=	将左边的值加上右边的值赋给左边的变量
−=	将左边的值减去右边的值赋给左边的变量
*=	将左边的值乘以右边的值赋给左边的变量
/=	将左边的值除以右边的值赋给左边的变量
%=	将左边的值对右边取余数赋给左边的变量
.=	将左边的字符串连接到右边

例如，“$a+=$b”等价于“$a=$a+$b”，其他赋值运算的等价关系可依此类推。赋值运算可以让程序更精简，增加程序的执行效率。

4. 位运算符

PHP 中的位运算符有 6 个，常用于二进制的运算场合，其具体含义如表 6-4 所示。

表 6-4　PHP 的位运算符

符　号	意　义	符　号	意　义
&	按位与	<<	按位左移
|	按位或	>>	按位右移
^	按位异或	～	按位取反

其中，“～”是单目运算符，其他的都是双目运算符。与、或、异或和取反运算的运算规则如下：

0&0=0，0&1=0，1&0=0，1&1=1（与：有假就假，都真才真）。

0|0=0，0|1=1 1|0=1，1|1=1（或：有真就真，都假才假）。

0^0=0，0^1=1，1^0=1，1^1=0（异或：相等为假，不等为真）。

～0=1，～1=0

注意：在对十进制进行位运算时要先转为二进制，然后按上述规则进行计算。

5. 逻辑运算符

逻辑运算通常用来测试值为真还是假。逻辑运算经常用在条件判断和循环处理中，在条件判断中用来判断条件是否满足在循环中用来判断是否该结束循环或继续执行循环体。逻辑运算符的具体含义如表 6-5 所示。

表 6-5　PHP 的逻辑运算符

符　号	意　义	符　号	意　义
&& (and)	逻辑与	xor	逻辑异或
|| (or)	逻辑或	! (not)	逻辑非

逻辑运算的真值表如表 6-6 所示。

表 6-6 逻辑运算真值表

$x	$y	$x && $y	$x || $y	$x xor $y	!$x
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

注意：不要将逻辑与、或、非（&&、||、!）同按位与、或、取反（&、|、~）相混淆。

6. 比较运算符

比较运算符和逻辑运算符的用法差不多，通过比较大小来测试值为真还是假，经常用在条件判断和循环处理中，在条件判断中用来判断条件是否满足，在循环中用来判断是否该结束循环或继续执行循环体。比较运算符的具体含义如表 6-7 所示。

表 6-7 PHP 的比较运算符

符 号	意 义	符 号	意 义
<	小于	==	等于（不包括类型）
>	大于	===	完全相等（包括类型）
<=	小于或等于	!=	不等于
>=	大于或等于		

7. 其他运算符

除了上述的运算符号之外，还有一些难以归类的、用于其他用途的运算符，其具体含义如表 6-8 所示。

表 6-8 PHP 的其他运算符

符 号	意 义	符 号	意 义
$	用于定义变量	–>	引用对象的方法或者属性
&	变量的地址(加在变量前引用变量)	=>	用于给数组元素赋值
@	屏蔽错误信息(加在函数前)	? :	三目运算符，条件表达式

其中比较特殊的是三目运算符“?:”，例如：

```
(expr1) ? (expr2) : (expr3);
```

表示如果 expr1 的运算结果为 true，则执行 expr2；否则执行 expr3。实际上它与 if…else 语句类似，但可以让程序更为精简和高。

此外，还有用于新对象的定义符 new、用于数组下标引用的方括号“[]”、表示结合性的大括号“{}”等。

6.5.2 运算符的优先级与结合性

运算符的优先级指定了两个表达式绑定得有多“紧密”。例如，表达式 1+5*3 的结果

是 16 而不是 18，这是因为乘号（“*”）的优先级比加号（“+”）高。必要时可以用括号来强制改变优先级，如(1+5)*3 的值为 18。如果运算符优先级相同，则使用从左到右的顺序。

表 6-9 所示为按优先级从高到低列出了所有的运算符。同一行中的运算符具有相同的优先级，此时它们的结合方向决定求值顺序。

表 6-9　PHP 运算符的优先级与结合性

优　先　级	结合方向	运　算　符	附 加 信 息
1（最高）	非结合	new	new
2	自左向右	[ ]	array ()
3	非结合	++　––	自增/自减运算符
4	非结合	!　~　– (int) (float) (string) (array) (object) @	类型转换
5	自左向右	*　/　%	算术运算符
6	自左向右	+　–　.	算术运算符和字符串运算符
7	自左向右	<<　>>	位运算符
8	非结合	<　<=　>　>=	比较运算符
9	非结合	==　!=　===　!==	比较运算符
10	自左向右	&	位运算符和引用
11	自左向右	^	位运算符
12	自左向右	|	位运算符
13	自左向右	&&	逻辑运算符
14	自左向右	||	逻辑运算符
15	自左向右	? :	条件运算符
16	自右向左	=　+=　–=　*=　/=　.=　%=　&=　|=　^=　<<=　>>=	赋值运算符
17	自左向右	and	逻辑运算符
18	自左向右	xor	逻辑运算符
19	自左向右	or	逻辑运算符
20	自左向右	,	多处用到

6.5.3　表达式

表达式就是由操作数、操作符以及括号等所组成的合法序列。简单地说，PHP 中的常量或变量通过运算符连接后就形成了表达式，如“$a=1”为一个表达式。表达式也有值，如表达式“$a=1”的值就是 1。

根据表达式中运算符类型的不同又可以把表达式分为算术表达式、字符串连接表达式、赋值表达式、位运算表达式、逻辑表达式、比较表达式等。

比较表达式和逻辑表达式是常见的表达式，这种表达式的值只能是真或假，在程序的流程控制中，会大量使用这两种表达式。

6.6 流程控制语句

几乎在所有的编程语言中，程序都由 3 种基本的结构组成，即顺序结构、分支结构和循环结构。

程序是由若干语句组成的，如果程序中语句的执行顺序是从上到下依次逐句执行的，那么这个程序的结构就是顺序结构，在这种结构中没有分支和反复，这也是最简单、最常见的流程结构，这里就不再过多地讨论。

PHP 中提供了 4 条流程控制语句实现分支结构和循环结构。

① if…else…为条件语句。

② switch 为分支选择语句。

③ do…while…为循环语句。

④ for 为循环语句。

此外，PHP 还提供了 break 语句和 continue 语句，用以跳出分支结构或循环结构，下面对这些语句逐一进行详细的介绍。

6.6.1 分支控制语句

1. if…else…语句

if…else…语句共有 3 种基本结构，此外每种基本结构还可以嵌套另外两种结构，而且还允许多级嵌套。

（1）只有 if 的语句

这种结构可以当作单纯的判断，可解释成“若某条件成立则去做什么事情”，其语法如下：

```
if(expr)
{
            statement
}
```

其中的 expr 为判断的条件表达式，通常都为比较表达式或逻辑表达式，而 statement 表示符合条件时执行的语句，若 statement 所代表的语句只有一行，可以省略大括号{}。如果 expr 为真，则执行 statement 语句或语句体。

（2）if…else…语句

这种结构可解释成“若某条件成立则去做什么事情，否则去做另一件事情”，其语法如下：

```
if(expr)
{
            statement1
}else{
            statement2
}
```

如果 expr 为真，则执行 statement1 语句或语句体；否则执行 statement2 语句或语句体。

（3）包含 else if 的语句

前面的两种分支结构只能实现二路分支，用包含 else if 的语句则可以实现多路分支，其语法如下：

```
if(expr1)
{
            statement1
}
else if (expr2)
{
            statement2
}
else if…
else
{
            statementn
}
```

当 expr1 为真时，执行 statement1 语句或语句体；否则转入 expr2 的判断。如果 expr2 为真，则执行 statement2 语句或语句体。依此类推，如果所有的 expr 表达式都不为真，则执行 statementn 语句或语句体。程序 6-19.php 为 if…else…语句应用示例。

程序 6-19.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-19.php: if…else…的应用-->
<head>
  <title>6-19.php</title>
</head>
<body>
  <?php
   //本程序测试时，可更改测试服务器的系统时间查看效果
   echo "今天是：";
   if (date("D")=="Sat")
      echo "周六，下午 3:00 参加新技术讲座。<br>";
   else if (date("D")=="Sun")
       {
       echo "今天是周日，晚 7:00 召开班例会。";
       }
  ?>
</body>
</html>
```

程序6-19中的date函数是格式化服务器的时间函数，date("D")返回服务器时间的星期时间中的英文的前 3 个字符。当系统时间不为星期六时，其运行结果如图 6-19 所示。

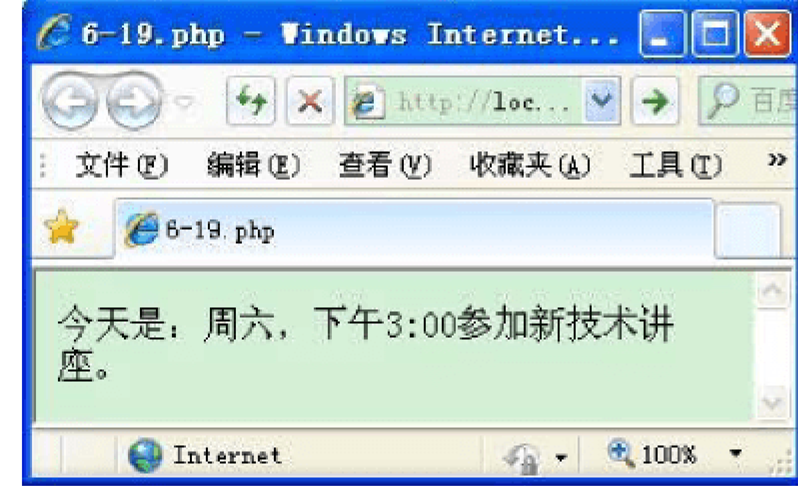


图 6-19　程序 6-19.php 的运行结果

在上述 3 种基本结构中，如果在 statement 语句体中还有 if…else…语句，就构成 if…else…语句的嵌套。

2. switch 语句

嵌套的 if…else…语句可以处理多分支流程，但使用起来比较烦琐而且也不太清晰，为此 PHP 中又引进了 switch 语句。其语法如下：

```
switch (expr)
{
        case expr1:
            statement1;
            break;
        case expr2:
            statement2;
            break;
        ⋮
        default:
            statementN;
            break;
}
```

其中的 expr 为条件，通常是变量名称。而 case 后的 exprN，通常表示变量的值。冒号后则为符合该条件要执行的语句。一定要注意 break 的作用为退出 switch 结构，千万不能省略不写。在设计 switch 语句时，要将出现几率最大的条件放在最前面，最少出现的条件放在最后面，可以增加程序的执行效率。具体使用如程序 6-20.php 所示。

程序 6-20.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-20.php: switch 的应用-->
<head>
 <title>6-20.php</title>
</head>
<body>
  <?php
     //本程序测试时，可更改测试服务器的系统时间查看效果
    switch (date("D")) {
      case "Mon":
          echo "今天星期一";
          break;
      case "Tue":
          echo "今天星期二";
          break;
      case "Wed":
          echo "今天星期三";
          break;
      case "Thu":
          echo "今天星期四";
```

```
            break;
        case "Fri":
            echo "今天星期五";
            break;
        default:
            echo "今天休息日";
            break;
    }
  ?>
</body>
</html>
```



程序 6-20.php 的运行效果如图 6-20 所示。

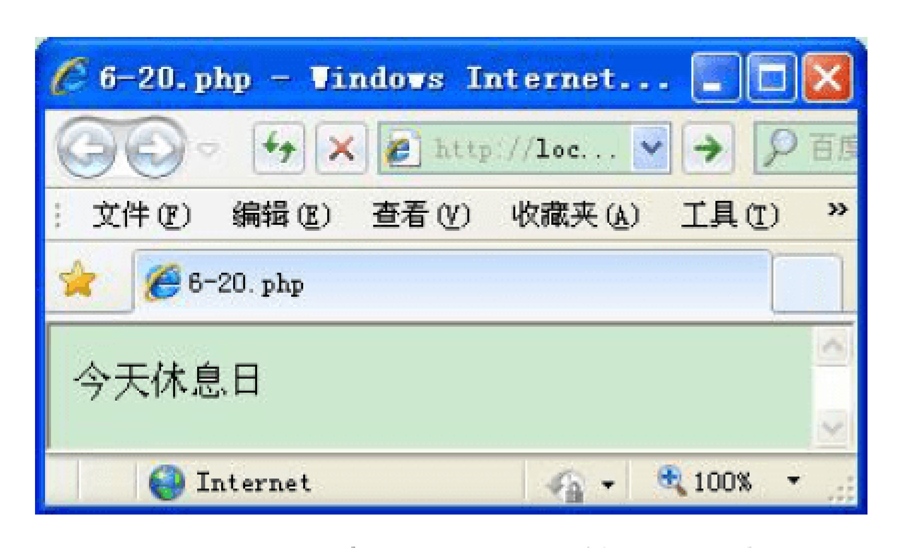


图 6-20 程序 6-20.php 的运行结果

6.6.2 循环控制语句

1. do…while…语句

在 PHP 中，do…while…循环语句有两种结构，一种只有 while…部分，另一种是 do…while…两部分都有。

（1）只有 while…部分的语句。

其语法如下：

```
while (expr)
{
            statement
}
```

（2）do…while…两部分都有的语句。

其语法如下：

```
do
{
            statement
} while (expr);
```

其中的 expr 为判断的条件，通常为逻辑表达式或比较表达式。而 statement 为符合条件的执行部分程序，若程序只有一行，可以省略大括号{}。

两种结构的区别在于，前者是先判断条件再执行语句；后者是先执行语句再判断条件。对于 expr 开始为真的情况，两种结构是没有区别的。如果 expr 开始为假，则前者不执行任何语句就跳出循环，而后者仍要执行一次循环才能跳出循环。程序 6-21.php 通过用级数求和近似无理数 e 的方法演示了 while 循环的应用。

程序 6-21.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-21.php: while 循环-->
<head>
   <title>6-21.php</title>
```

```
</head>
<body>
  <?php
   $i=0;
   $t=1;
   $e=0;
   while  ($t>0.000001){
   $e=$e+$t;
  $i++;
   $t=$t/$i;
   }
   echo 'e='.$e;
   ?>
</body>
</html>
```

其运行结果如图 6-21 所示。

程序 6-22.php 则演示了 do…while 循环的应用。

程序 6-22.php

```
<?php
        header("Content-type:image/png");
        $image=imagecreate(200,120)or die("无法创建图像！");
        imagecolorallocate($image,250,250,250);
        $color=imagecolorallocate($image,250,0,0);
         $radius=20;
        do{
         imageellipse($image,100,60,$radius,$radius,$color );
         $radius =$radius+20;
       }while($radius<100);
        imagepng($image );
        imagedestroy($image);
  ?>
```

程序运行结果如图 6-22 所示。



图 6-21 程序 6-21 运行结果

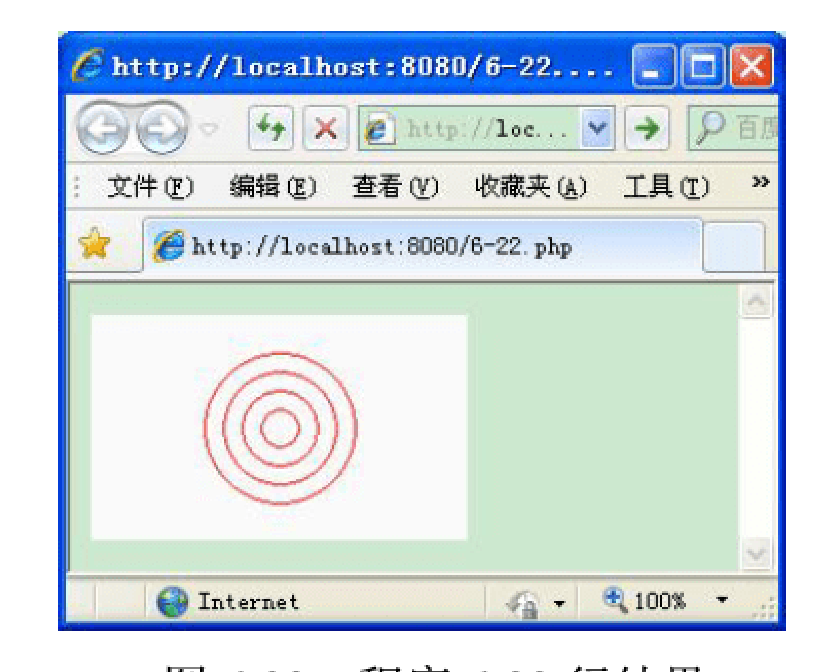


图 6-22 程序 6-22 行结果

2. for 语句

for 语句仅有单纯的一种，没有其他变化，但同时也是最复杂、功能最强大的循环语句，任何 while 循环和 do…while…循环都可以用 for 循环代替。for 语句的语法如下：

```
for(expr1;expr2;expr3)
{
    statement
}
```

其中，expr1 为条件的初始值；expr2 为判断的条件，通常都是用比较表达式或逻辑表达式作为判断的条件；expr3 为执行 statement 后要执行的部分，即循环步长，可以用来改变条件，供下次的循环判断，如将循环变量加 1、减 1 等。statement 为符合条件后执行的语句或语句体，若 statement 只有一条语句组成，则可以省略大括号{}。在明确循环次数的情况下往往选择用 for 循环语句。程序 6-23.php 演示了 for 循环的应用。

程序 6-23.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-23.php: for 循环-->
<head>
 <title>6-23.php</title>
</head>
<body>
  <?php
   for($i=1;$i<5;$i++) {
    for($j=1;$j<=10;$j++)
    {
      echo "*";
    }
    echo "<br>";
   }
  ?>
</body>
</html>
```



图 6-23　程序 6-23 运行结果

程序运行结果如图 6-23 所示。

3. break 和 continue 语句

在 switch 和 for 语句的例子中都用到了 break 语句，它的作用就是跳出整个 switch 分支结构或 for 循环结构，执行其下面的语句。而 continue 经常用在 for 或 do…while…循环语句中，表示跳出本次循环，继续进入下一次循环。这也是 break 和 continue 的主要区别，如程序 6-24.php 所示。

程序 6-24.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-24.php: break/continue 的应用-->
<head>
  <title>6-24.php</title>
</head>
<body>
 <?php
```

```
 echo "使用 break 的输出效果：<br>";
 $i=0;
 while ($i<10) {
   if ($i%2==1) {
      break;
   }
   echo $i;
   $i++;
 }
 echo "<br>使用 contine 的输出效果：<br>";
 for($i=0;$i<10;$i++){
   if ($i%2){
       continue;
       }
   echo $i." ";
     }
 ?>
</body>
</html>
```

程序 6-24.php 运行结果如图 6-24 所示。



图 6-24　程序 6-24.php 的运行结果

6.7　PHP 网页通信

网站是网页的集合，一个网站中会有众多的网页——PHP 文件，每个文件能够完成的功能有限。为了能够完成和客户的交互，需要多个 PHP 文件共同协作。因此，掌握 PHP 网页的通信技术是非常重要的。本小节将介绍主要的网页间通信的方法，包括使用表单，使用 Cookie，使用 Session 以及页面跳转等内容。

6.7.1　使用表单

表单是用户和程序间交互的重要工具，是采集和提交用户输入的主要地方。表单允许用户以标准格式向服务器提交数据，数据被传送给某个 PHP 文件并被进行相应处理。提交数据的方式有 Get 方法和 Post 方法两种。6.3.4 节对 PHP 的预定义变量做了简单介绍，其中的$_GET 和$_POST 分别用来接收以 Get 和 Post 方法提交数据的预定义变量。本节将详

细说明它们的用法。

POST 方法是在 HTTP 请求中嵌入表单数据；GET 方法则将表单数据附加到请求该页的 URL 中。由第 2 章知道，表单标记<form>的属性 method 的取值有两个，取为 post 时，表示以 post 方法提交；取为 get 时，表示以 get 方法提交。属性 action 指定数据提交给哪个文件。例如：

```
<form method= "get" action= "test.php">
```

表示用 get 方法提交数据，数据提交给文件 test.php，可在这个文件中接收提交的数据并进行处理。

表单中可以包含很多的控件，如文本框、单选按钮、复选框、文件域、滚动文本框、按钮等。接收表单数据即指获取表单控件的 value 属性的值。不同的控件可以设置不同的 name 属性，在接收数据时根据 name 属性确定是哪个控件的值。不同的控件设置 value 属性的方式也不一样。例如，单选按钮可能由多个选项组成，这些选项的 name 属性值都相同时表示这些选项属于同一个表单控件，每个选项都有一个 value 值，接收控件的值后可以根据这个 value 值判断用户选择了哪个选项。又例如，复选框控件可以使用户选择多个选项，复选框中选项的 name 属性值都设置为相同，并且设置为数组的形式，如 name="testpaper[]"，而每个选项都有一个 value 值，接收数据时接收到的是一个数组，数组中保存了用户选择的选项，遍历数组的值就可以确定用户选择了哪些选项。下面的程序 6-25.php 演示了用 get 方法提交数据的情况。

程序 6-25.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-25.php：get 方法传递表单数据传递-->
<head>
  <title>6-25.php</titleE>
</head>
<body>
  <?php
  $tag=@$_GET["tag"];
  if ($tag==1){
      $add1=$_GET["add1"];
      $add2=$_GET["add2"];
      }else{
          $add1=0;
          $add2=0;
          }
  $sum=$add1+$add2;
  ?>
请输入两个加数：
<form name="form1" method="get" action="#">
<!--下面是一个隐藏表单，接收后用以来判断是提交前的页面还是提交后的页面-->
  <input type="hidden" name="tag" size="4" value="1">
 <input type="text" name="add1" size="4" value="<?php echo $add1;?>">
```

```
  +
  <input type="text" name="add2" size="4" value="<?php echo $add2;?>">
  =
  <?php echo $sum;?><br>
<br><input type="submit" name="btn1" value="计算">
<input type="reset" name="btn2" value="重置">
</form>
</body>
</html>
```

程序运行结果如图 6-25 所示。

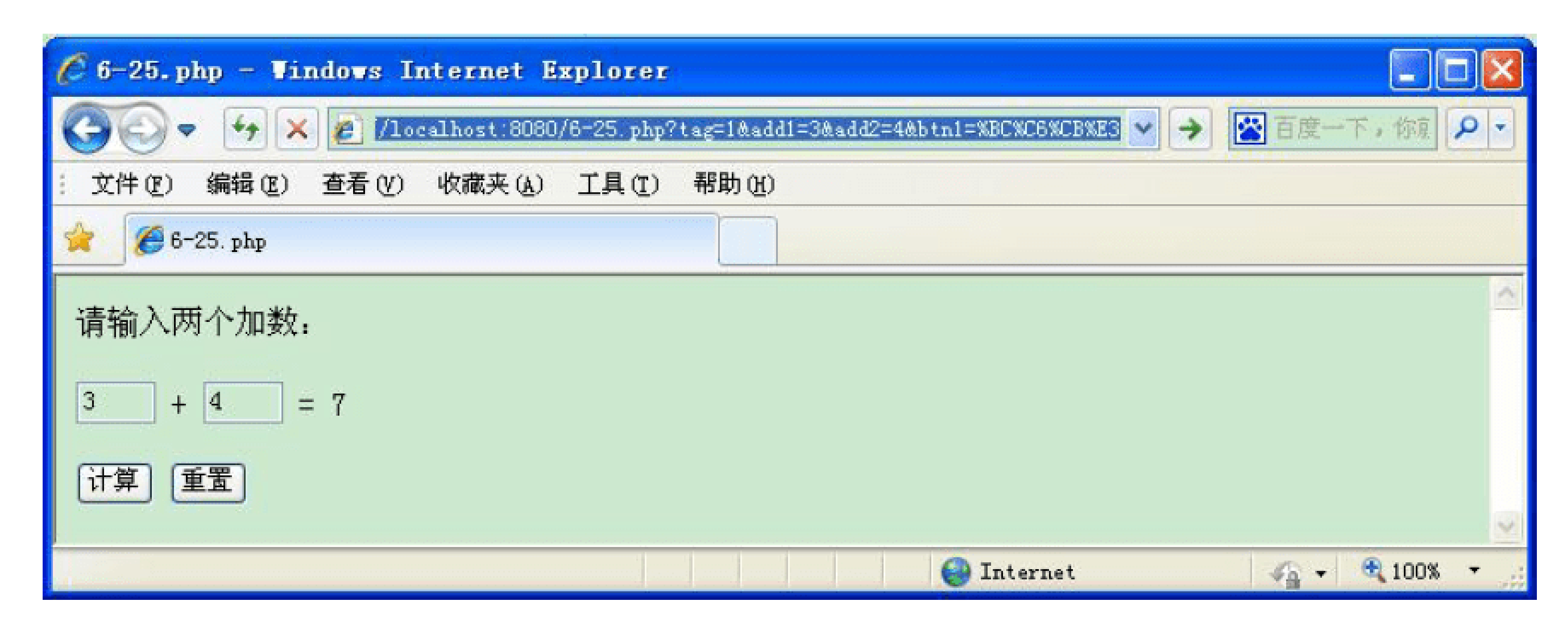


图 6-25 程序 6-25.php 的运行结果

图 6-25 是在输入两个加数并单击“计算”按钮提交后的运行结果。注意此时地址栏中的变化。在“http://localhost:8080/6-25.php?tag=1&add1=3&add2=4&btn1=%BC%C6%CB%E3”中，问号后面 tag=1&add1=3&add2=4，分别记录了 name 是 tag、name 是 add1 和 name 是 add2 的三个表单控件的值，上一个值和下一个值之间用“&”号连接。name 是 btn1 的控件（即“计算”按钮）的值则是经过编码后的结果。也就是说，用 get 方法提交的数据，是附加在地址栏中传递的，这种方法传递数据的规模较小，如果不经过编码，所传递的数据是可见的，当然也是不安全的。下面的程序 6-26 演示了用 post 方法传递数据的情况。

程序 6-26(1).php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-26.php: post 方法传递表单数据(1)-->
<html>
<head>
<title>6-26(1).php</title>
</head>
<body>
<form action="6-26 (2).php" method="POST" name="myform">
  <center>
<table  id="table1"  width="508"  cellspacing="0"  cellpadding="0"
border="1" bordercolorlight="#000000" bordercolordark="#FFFFFF">
  <tr>
  <td height="22" colspan="3" align="center"><b>自 我 介 绍</b></td>
  </tr>
```

```
<tr>
<td height="22">名字：</td>
<td height="22"><input type=text name="name" size=15 maxlength=8 onBlur="namecheckdata();"></td>
<td height="22" width="105">不能为空，只能是字母</td>
</tr>
<tr>
<td height="22">性别：</td>
<td height="22">男<input type="radio" name="sex" value="男" checked>  女<input type="radio" name="sex" value="女"></td>
<td height="22" width="105"> </td>
</tr>
<tr>
<td height="24">年龄：</td>
<td height="24"><input type=text name="age" size="10" maxlength="8"></td>
<td width="105" height="24">不能为空，在 0 到 100 之间</td>
</tr>
<tr>
<td>个人密码：</td>
<td><input type=password name="password" size=15 maxlength=8></td>
<td width="105">不能为空，4 到 10 位数</td>
</tr>
<tr>
<td>确认密码：</td>
<td><input type=password name="password1" size=15 maxlength=8></td>
<td width="105">与个人密码相同</td>
</tr>
<tr>
<td>你的爱好：</td>
<td>看书<input type="checkbox" name="like1" value="看书"> 足球<input type="checkbox" name="like2" value="足球" checked> 音乐<input type="checkbox" name="like3" value="音乐">    山<input type="checkbox" name="like4" value="  山"><!--多选按钮--></td>
<td width="105"> </td>
</tr>
<tr>
<td>你最喜欢的颜色：</td>
<td><select name="color" size="1">
    <option name="red">红色
    <option name="green" selected>绿色
    <option name="blue">蓝色
    </select></td>
<td width="105"> </td>
</tr>
<tr>
<td>个人介绍：</td>
<td><textarea cols="30" rows="5" name="jieshao"></textarea></td>
<td width="105">不能为空</td>
</tr>
```

```
 <tr>
 <td colspan="3" align="center"><input name="submit" type="submit"
value="提交">  <input name="reset" type="reset" value="重置
"></td></tr>
</table></center>
</form>
</body>
</html>
```

程序运行结果如图 6-26 所示。

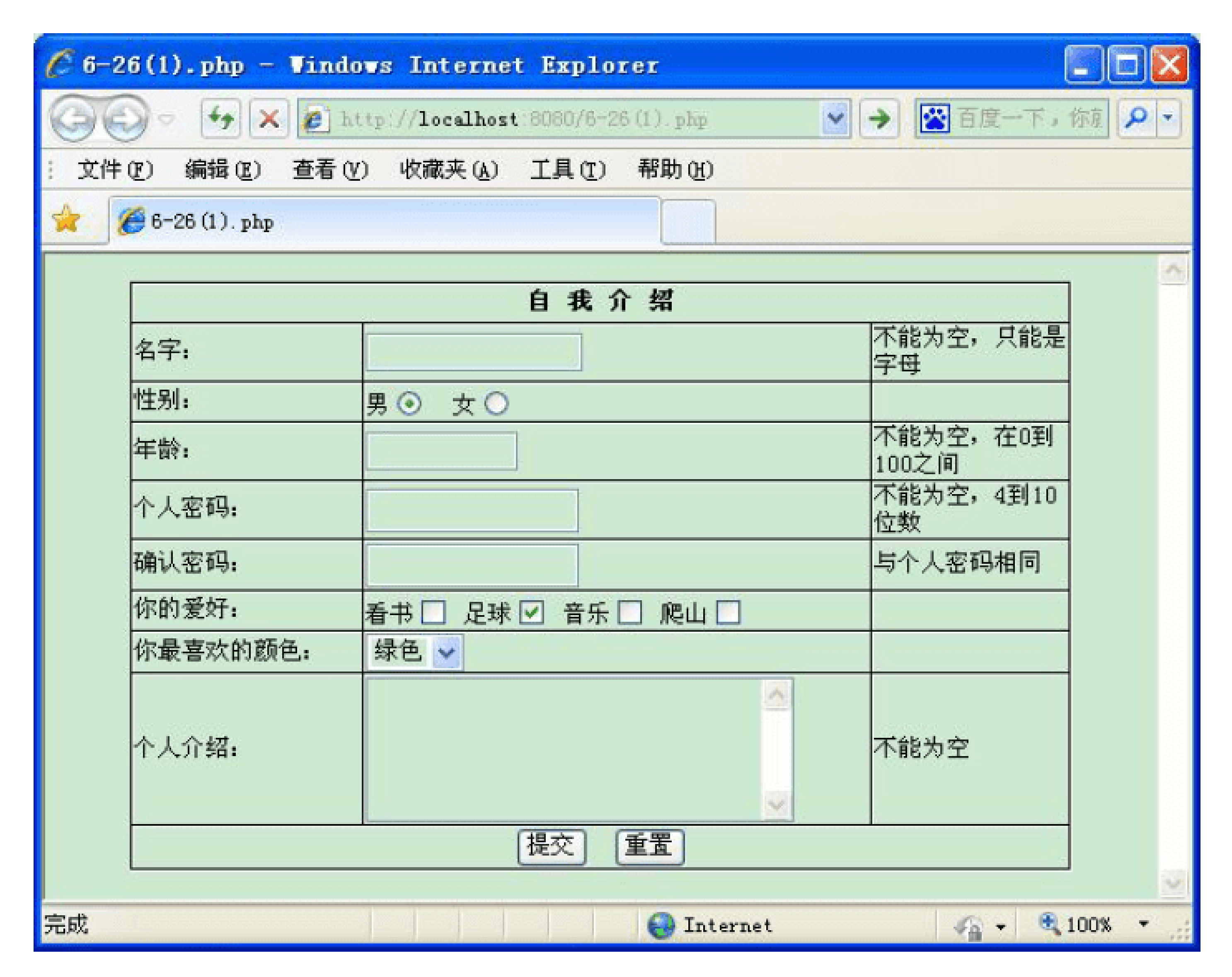


图 6-26　程序 6-26(1).php 的运行结果

填写表单，提交数据，数据将提交程序 6-26(2).php。程序 6-26(2).php 清单如下。

程序 6-26(2).php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 6-26.php: post 方法传递表单数据(2)-->
<html>
<head>
<title>6-26(2).php</title>
</head>
<body>
<div align="center">
<?php
  //接收 bxbz5.htm 中的数据
  $name=$_POST["name"];
  $sex=$_POST["sex"];
  $age=$_POST["age"];
  $password=$_POST["password"];
  $password1=$_POST["password1"];
```

```
  $like1=$_POST["like1"];
  $like2=$_POST["like2"];
  $like3=@$_POST["like3"];
  $like4=@$_POST["like4"];
  $color=$_POST["color"];
  $jieshao=$_POST["jieshao"];
  //验证表单数据的合法性
  if($name==""){ echo "===名字不能为空，请
<a href=javascript:history.go(-1)>返回</a>重新输入!===";
    exit;
     }
  else if($age<1 or $age>100){ echo "===年龄不再 0 到 100 之间，请
<a href=javascript:history.go(-1)>返回</a>重新输入!===";
  exit;
     }
else if (strlen($password)<4 or strlen($password)>10 or strlen($password1)<4 or strlen($password1)>10 or $password!=$password1){
echo "===两次密码长度均在 4-10 之间且要相等，请<a href=javascript: history.go(-1)>返回</a>重新输入!===";
      exit;
         }
       else if($jieshao==""){ echo "===个人介绍不能为空，请
<a href=javascript:history.go(-1)>返回</a>重新输入!===";
        exit;
         }
  ?>  的自我介绍信息提交成功，请核对!
</div>
<table width="508" border="1" align="center" cellpadding="0" cellspacing="0" bord ercolorlight="#000000" bordercolordark="#FF2146" id="table1">
    <tr><td width="113" height="22">名字：</td>
    <td width="389" height="22"><?php echo $name?></td>
    </tr><tr>
    <td height="22">性别：</td><td height="22"><?php echo $sex?></td>
    </tr><tr>
    <td height="24">年龄：</td><td height="24"><?php echo $age?></td>
    </tr><tr>
    <td>个人密码：</td><td>已设置</td>
    </tr><tr>
    <td>你的爱好：</td>
    <td>
    <?php echo $like1."  ".$like2."  ".$like3."  ".$like4."  "?></td>
    </tr><tr>
    <td>你最喜欢的颜色：</td><td><?php echo $color?></td>
    </tr><tr>
    <td>个人介绍：</td><td><?php echo $jieshao?></td>
  </tr></table>
 </body>
</html>
```

程序运行结果如图 6-27 所示。

图 6-27　程序 6-26(2).php 的运行结果

Post 方法提交数据，能够传送较大规模的数据量，并且数据不是附加在地址栏中，对用户不可见，具有很好的数据安全性。

6.7.2　页面跳转

页面跳转是网站开发中经常遇到的问题，通过页面跳转，可以在相关 PHP 文件中传递数据，实现设计功能。本节介绍几种常用的页面跳转方法。

1. 提交表单

最常用的跳转页面的方法是通过提交表单，跳转到相应的网页。做法是将<form>标记的 action 属性设置为要跳转到的页面，提交表单后就跳转到该页面。例如：

```
<form method="post" action="index.php">
<input type="text" name="text">
<input type="submit" name="bt" value="提交">
</form>
```

这种方法在 6.7.1 节已有详细应用。

2. 使用 XHTML 标记

使用 XHTML 的超链接标记<a>也能够实现跳转页面的功能，例如：

```
<?php
echo "<a href='index.php?id=3&class=4'>单击超链接</a>";
echo "<a href='http://www.sohu.com'>转到搜  网站</a>";
?>
```

除了使用超链接标记<a>以外，另外一种方法是使用<meta>标记，例如：

```
<meta http-equiv="refresh" content="3;url=index.php">
```

以上代码的作用是 3 秒之后跳转到 index.php 页面。content 属性中数字 3 表示 3 秒之后跳转，设置为 0 则表示立即跳转，url 选项可以指定要跳转到的页面。如果要刷新本页面，则可以省略 url 选项，代码如下：

```
<meta http-equiv="refresh" content="3">
```

3. 使用 header 函数

header 函数的作用是向浏览器发送正确的 HTTP 报头，报头指定了网页内容的类型、页面的属性等信息。header 函数的功能很多，其中之一便是跳转页面。如果 header 函数的参数为“Location: xxx”，页面就会自动跳转到 xxx 指向的 URL 地址。例如：

```
<?php
$var1="abc";
$var2="da";
if($var1==$var2)
{
   header("Location: http://www.sohu.com");//跳转到搜  主页面
}
else
   header("Location: jczd.php");          //跳转到工作目录的jczd.php 页面
?>
```

4. 使用按钮

使用按钮跳转页面，只需要在按钮控件的 onclick 方法中设置执行的代码即可，例如：

```
<?php
 echo '<input type="button" name="btn" value="页面跳转"
onclick= "location= \'index.php\'">';
?>
```

5. 使用客户端脚本

在 PHP 中可以使用 JavaScript 跳转到页面，例如：

```
<?php
echo "<script>if(confirm('确认跳转页面？')) ";
echo "window.location='index.php'</script>";
//上面一句也可写做 echo "location.href='index.php'; </script>";
?>
```

6.7.3　使用 Cookie

当到图书馆借书时，要带上身份证明，如一卡通或借书证。当登录某个网站时，也要带一个身份证明，那就是 Cookie。Cookie 是保存在客户本地计算机上的一小段有关客户信息的文本数据。不论访问哪个网站，都是通过浏览器访问。浏览器访问网站的过程大致如下：

（1）当在浏览器地址栏里输入目标网站的 URL 时，就确定了需要访问的网站，如果连接成功，浏览器向 Web 服务器发送访问该网站的请求。

（2）如果 Web 服务器接受了请求，Web 浏览器将对给定的网站在本机上查找现存的相关 Cookie。

（3）如果找到了相关 Cookie，浏览器就会以 HTTP 首部的形式将 Cookie 中的名称-数值对发送给服务器。例如：

```
GET http://sohu.com/ HTTP/1.1
Accept: application/x-ms-application,image/jpeg,application/xaml+xml, [...]
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; [...]
```

```
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
Host: sohu.com
Cookie: datr=1265876274-[...]; locale=en_US; lsd=WW[...]; c_user=2101[...]
```

请求中包含了浏览器存储的该域名的 Cookie。

（4）如果浏览器在本机上找不到相应的 Cookie，就会通知服务器 Cookie 缺失，服务器将会为请求连接的客户产生新的 ID，并将含有名称-数值对的 Cookie 发送到发出请求的 Web 浏览器上，浏览器接着就把这个 Cookie 存储在本机硬盘上。

从客户端发送的 Cookie 都会被 PHP 5 自动加入预定义变量$_COOKIE 中，这是一个全局数性的数组变量。如果希望对一个 Cookie 变量设置多个值，则需在 Cookie 的名称后加“["值名称"]”符号。

1. Cookie 的设置和读取

函数语法：

```
bool setcookie(string name[, string value[, int expire[, string path[, string
domain[, bool secure[, bool httponly]]]]]])
```

函数说明：

setcookie()定义一个和其余的 HTTP 标头一起发送的 Cookie。和其他标头一样，Cookie 必须在脚本的任何其他输出之前发送（这是协议限制）。这需要将本函数的调用放到任何输出之前，包括<html>和<head>标签以及任何空格。如果在调用 setcookie()之前有任何输出，本函数将失败并返回 FALSE。除了 name 外，其他所有参数都是可选的。可以用空字符串（""）替换某参数以跳过该参数。因为参数 expire 是整型，不能用空字符串跳过，可以用零（0）代替。

- name：Cookie 的名字。
- value：Cookie 的值。
- expire：Cookie 过期的时间，常用 time 函数再加上秒数来设定。
- path：Cookie 在服务器端的有效路径；设为'/'，在整个 domain 内有效；设为'/foo/'，在 domain 下的/foo/目录及其子目录内有效。默认值为设定 Cookie 的当前目录。
- domain：Cookie 的有效域名。
- secure：指明 Cookie 是否仅通过安全的 HTTPS 连接传送。当设成 TRUE 时，Cookie 仅在安全的连接中被设置。默认值为 FALSE。

例如：

```
<?php
//文件 a.php，用于设置 cookie
$str1="Python 语言很好！";
if(!isset($_COOKIE['zjx']))         //判断 Cookie 是否存在
setcookie("zjx",$str1,time()+360); //设置 Cookie 的名、值和存活时间
echo '<a href="b.php">输出</a>';     //设置读取 Cookie 的文件
?>
```

Cookie 设置后，便可以在其他页面通过$_COOKIE 预定义变量取得其值。因为 Cookie 不会在设置它的页面生效，在其到期以前，需要通过另外的页面访问。

```
<?php
//文件 b.php 用于显示文件 a.php 中设置的 cookie，两个文件位于同一目录下
echo $_COOKIE ["zjx"];
?>
```

2. Cookie 的删除

用 setcookie 函数设置 Cookie 的过期时间为过去的时间，值为空即可清除 Cookie 的内容。例如：

```
<?php
//文件 c.php
$str1="Python 语言";
if(!isset($_COOKIE['zjx'])){
setcookie("zjx",$str1,time()+3600);
echo"请按 F5 键刷新页面，保存 cookie。";    }
else
echo "刚设定的名为 zjx 的 cookie 的值是：".$_COOKIE ['zjx'];
echo "<br>";
echo '<a href="d.php">删除 cookie</a>';
?>
```

删除 Cookie 的文件 d.php 如下：

```
<?php
//方法一：
setcookie("zjx","", time()-3600);
if(isset($_COOKIE ["zjx"]))
echo $_COOKIE ["zjx"];
else
echo "Cookie 已经删除！";
?>
```

注意：当一个 Cookie 被删除时，它的值在当前页面仍然有效。

下面的程序用 Cookie 验证用户的登录信息。该实例包括两个程序，程序 6-27(1).php 是登录页面，通过这个页面提交数据；程序 6-27(2).php 接收 6-27(1).php 提交的数据并对提交的数据进行验证。

程序 6-27(1).php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--文件 6-27(1).php：COOKIE 实现用户登录的表单-->
<head>
  <title>6-27(1).php</title>
</head>
<body>
<form name="form1" method="post" action="6-27(2).php">
<table width="280" height="96" border="0" align="center" cellpadding="0"
cellspacing="1" bgcolor="#999999">
  <tr>
    <td colspan="2" align="center" bgcolor="#FFFFFF">用户登录</td>
  </tr>
```

```
  <tr>
    <td align="right" bgcolor="#FFFFFF">用户名:</td>
    <td align="left" bgcolor="#FFFFFF">
       <input type="text" name="user_name" size="12">
    </td>
  </tr>
  <tr>
    <td align="right" bgcolor="#FFFFFF">口令:</td>
    <td align="left" bgcolor="#FFFFFF">
    <input type="password" name="user_pw" size="12"></td>
  </tr>
  <tr>
    <td colspan="2" align="center" bgcolor="#FFFFFF">
    <input type="submit" name="Submit" value="提交"> 
    <input type="reset" name="Submit2" value="重置"></td>
  </tr>
</table>
</form>
</body>
</html>
```

程序运行结果如图 6-28 所示。



图 6-28 程序 6-27(1).php 的运行结果

在表单上填写数据，单击“提交”按钮，数据将被传送到程序 6-27(2).php 并被验证。

程序 6-27(2).php

```
<!--程序 6-27(2).php: COOKIE 数据页间传递-->
<?php
  setcookie("user_name", $_POST[user_name]);
  setcookie("user_pw", $_POST[user_pw]);
  ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <TITLE>6-27(2).php</TITLE>
</head>
<body>
```

```
<?php
if ($_POST["user_name"]=="zhangsan" && $_POST["user_pw"]=="php5")
echo "  喜  ，用户名和口令正确，登录成功！";
?>
</body>
</html>
```

程序运行结果如图 6-29 所示。



图 6-29　程序 6-27(2).php 的运行结果

6.7.4　使用 Session

6.7.3 小节介绍了 Cookie，实现了在网站各网页文件间数据的共享。但 Cookie 还存在一些问题：Cookie 数据存放在客户的浏览器上，别人可以分析存放在本地的 Cookie 并进行 Cookie　　；Cookie 的容量有限，一个浏览器能创建的 Cookie 数最多为 30 个，每个不能超过 4KB，每个 Web 站点能设置的 Cookie 总数不能超过 20 个；为了安全起见，有的客户可以通过设置浏览器选项来禁用 Cookie；还有些浏览器根本就不支持 Cookie。其实，除了 Cookie 以外，还有另一种实现网站各网页文件间数据的共享的机制，就是 Session。

Session 直接翻译成中文比较困难，一般都译成时域。在计算机专业术语中，Session 是指一个终端用户与交互系统进行通信的时间间隔，通常指从注册进入系统到注销退出系统之间所经过的时间。具体到 Web 中的 Session 指的就是用户在浏览某个网站时，从进入网站到浏览器关闭所经过的这段时间，也就是用户浏览这个网站所花费的时间。因此从上述定义中可以看到，Session 实际上是一个特定的时间概念。

需要注意的是，一个 Session 的概念需要包括特定的客户端，特定的服务器端以及不中断的操作时间。A 用户和 C 服务器建立连接时所处的 Session 同 B 用户和 C 服务器中建立连接时所处的 Sessions 是两个不同的 Session。

Session 成功的关键在于如何区分不同的客户端进程。在用户登录或访问一些初始页面时，服务器会为客户端分配一个 SessionID，这是一个加密的随机数字。服务器与客户端用这个 SessionID 来保持　此间的联系。

Session 存储于服务器端（默认以文件方式存储），用户在连接服务器时，服务器首先检查这个客户端的请求里是否已包含了一个 SessionID。如果已包含，则说明以前已经为此客户端创建过 Session，服务器就按照 SessionID 把这个 Session 检索出来使用。如果检索不到，或客户端请求不包含 SessionID，则为此客户端创建一个 Session 并且生成一个与此 Session 关联的唯一的 SessionID，这个 SessionID 将被在本次响应中返回给客户端并以 Cookie 的方式保存。

这样在交互过程中浏览器可以自动地按照规则把这个标识发送给服务器。用户提交页面时，会将这一 SessionID 提交到服务器端，存取 Session 数据。如果客户端禁用了Cookie，SessionID 会以 URL 的附加数据的方式传送给服务器。在 PHP 中实现 Session 的主要步骤包括，启动 Session，注册 Session 变量，使用 Session 变量，注销 Session 变量，销毁 Session。

1. 启动 Session

在 PHP 中有两种方法可以启动 Session。

（1）使用 session_start 函数，在其他任何代码之前先调用该函数。语法格式如下：

```
bool session_start(void)
```

该函数将检查是否有一个 SessionID，如果没有就创建一个，并且使其能够使用预定义变量$_SESSION 进行访问。如果 SessionID 已经存在，则将这个已经注册的 Session 变量载入以便使用。

（2）通过设置 php.ini 自动创建。在 php.ini 文件中，将 session.auto_start 选项激活（设置“session.auto_start=1”）即可。但这种方法有一个缺陷，会导致无法使用对象作为变量。

2. 注册 Session 变量

启动 Session 后，Session 变量会保存在预定义变量$_SESSION 中，这是一个数组，可以以直接定义数组单元的方式来定义一个会话变量，格式如下：

```
$_SESSION["键名"]="值";
```

Session 变量定义后被记录在服务器中，并对该变量的值进行跟踪，直到会话结束或手动注销该变量。

3. 使用 Session 变量

注册 Session 变量后，就可以使用它了。首先要使用 session_start 函数启动一个会话。之后就可以使用$_SESSION 数组访问该变量了。例如：

```
<?php
session_start();
if(isset($_SESSION["name"]))
{
   echo $_SESSION["name"];
}
else
   echo "会话变量未注册";
?>
```

4. 注销 Session 变量

Session 变量使用完后，删除已经注册的会话变量以减少对服务器资源的占用。删除会话变量使用 unset 函数，语法格式如下：

```
void unset(mixed $var [, mixed $var [, $... ]])
```

说明：$var 是要销毁的变量，可以销毁一个或多个变量。

例如：

```
<?php
$var="hello";
session_start();
```

```
$_SESSION["var"]=$var;                    //注册会话变量
unset($_SESSION["var"]);                  //删除会话变量
if(!isset($_SESSION["var"]))              //判断是否存在会话变量
   echo "删除成功";
?>
```

要一次销毁所有的会话变量，可以使用以下语句：

```
session_unset();
```

5. 销毁 Session

使用一个会话后，要注销所有的会话变量，然后再调用 session_destroy 函数销毁会话，语法格式如下：

```
bool session_destroy ( void )
```

该函数将删除会话的所有数据并清除 SessionID，关闭该会话。例如：

```
<?php
session_start();
session_destroy();
?>
```

Session 最典型的应用当属实现购物车。下面的程序 6-28 以实现购物车为例，演示了 Session 的应用。先建立一个购物车类 Cart，关于类的用法，请参看第 8 章。

程序 6-28

```
<?php
//购物车类，实际上就是一些函数集合
class Cart
{
 public function add($good,$num,$price)
 {
   $cart=$this->read_session();
   $key=$this->in_cart($good,$cart );
   if($key>0)
   {
     $cart [$key -1][1]+=$num;
     }
     else
     {
        $cart []=array($good,$num,$price);
     }
   $this->save_session($cart );
 } //end of add()
 public function del($good ,$num )
 {
   $cart=$this->read_cookie();
   $key=$this->in_cart($good,$cart );
   if($key>0)
   {
     if($cart [$key -1][1]>=$num) {
     $cart [$key -1][1]-=$num;
```

```
      $this->save_session($cart );
      return TRUE;
      }
      else
      {
        return FALSE;
      }
   }
  else
  {
    return FALSE;
  }


 }  //end of del()
  public function read_session()
  {
    $cart =array();
    if(isset($_SESSION ['cart']))
    {
      $array=explode("||",$_SESSION ['cart']);
      foreach($array  as $text)
      {
        $cart []=explode(",",$text);
      }
    }
    return $cart ;
  } //end of read_session()
  public function save_session($cart )
  {
     $array =array();
     foreach($cart  as $good )
     {
       if($good [1]>0)
       $array []=implode(",",$good );
       if(count($array )>0)
       //setcookie("cart",implode("||",$array ));
       $_SESSION['cart']=implode("||",$array);
       else
       //setcookie("cart","",time()-1);
       $_SESSION['cart']=null;
     }
  }//end of save_session()

  private function in_cart($good ,$cart )
  {
    foreach($cart as $key =>$value)
    {
      if(in_array($good ,$value))
      {
```

```
        return $key +1;
      }
    }
    return 0;
  } //end of  in_cart()

}//end of class
```

程序 6-28(1).html 是商品展示页面。

程序 6-28(1).html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http:// www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>6-28(1).php</title>
<style type="text/css">
div{margin:1px;
   margin-top:2px;
   margin-left:5px;
   border:1px solid #F36;
   width:470px;
   }
</style>
</head>
<body>
<p>    ，欢迎光临我的小店</p>
<div>
<form name="form1" action="6-28(2).php" method="post">
  商品名称：绿茶|<input name="cargo" type="hidden" value="绿茶"/>
  数量：<input name="num"  width="10" size="2"  value="1"/>
  单价：<input name="price"  width="6" size="6" readonly=
"readonly" value="260"/>
  库存：<input name="store"  width="6" size="3" readonly=
"readonly" value="大量"/>
<input name="act" type="hidden" value="buy"  />
<input type="submit" value="购  " />
</form>
</div>
<div>
<form name="form2" action="6-28(2).php" method="post">
  商品名称：军  |<input name="cargo" type="hidden" value="
军  "/>
  数量：<input name="num" value="1" width="10" size="2"
readonly="readonly"  />
  单价：<input name="price"  width="6" size="6" readonly=
"readonly" value="36"/>
  库存：<input name="store"  width="6" size="3" readonly=
"readonly" value="大量"/>
<input name="act" type="hidden" value="buy"  />
```

```
<input type="submit" value="购  " />
</form>
 </div>
 <div>
<form name="form3" action="6-28(2).php" method="post">
  商品名称：台  |<input name="cargo" type="hidden" value="台  "/>
  数量：<input name="num" width="10" size="2" value="1" />
  单价：<input name="price"  width="6" size="6" readonly=
"readonly" value="75"/>
  库存：<input name="store"  width="6" size="3" readonly=
"readonly" value="大量"/>
<input name="act" type="hidden" value="buy"/>
<input type="submit" value="购  " />
</form>
</div>
<div>
<form name="form4" action="6-28(2).php" method="post">
  商品名称：手机|<input name="cargo" type="hidden" value=
"手机"  />
  数量：<input name="num"  value="1"width="10" size="2"  />
  单价：<input name="price" width="6" size="6" readonly=
"readonly" value="730"/>
  库存：<input name="store"  width="6" size="3" readonly=
"readonly" value="大量"/>
<input name="act" type="hidden" value="buy"  />
<input type="submit" value="购  " />
</form>
  </div>
  <div>
<form name="form4" action="cart.php" method="post">
  商品名称：水杯|<input name="cargo" type="hidden" value=
"水杯"  />
  数量：<input name="num"  width="6" size="2" value="1" />
  单价：<input name="price"  value="30"width="6" size="6"
readonly="readonly"  />
   库 存 ：<input  name="store"   width="6"  size="3"
readonly="readonly" value="大量"/>
<input name="act" type="hidden" value="buy"  />
<input type="submit" value="购  " />
</form>
</div>
</body>
</html>
```

程序 6-28(2).php 为购物操作页面。

程序 6-28(2).php

```
<?php
session_start();
include("shopping_cart.php");
if($_POST['act']=="buy"&& isset($_POST['cargo'])&& isset($_POST['num'])&&
```

```
isset($_POST['price']))
{
$cart=new cart();
$cart->add(trim($_POST['cargo']),trim($_POST['num']), trim($_POST
      ['price']));
header("Location:6-28(3).php");
}
if($_POST['act']=="cancel" && isset($_POST['cargo'])&& isset($_POST
['num']))
{
$cart=new cart();
if(!$cart->del(trim($_POST['cargo']),trim($_POST['num'])))
{
    echo "取消本项商品失败！";
    }
else
{
    header("Location:6-28(3).php");
    }

}
if($_POST['act']=='cancel_all')
{
setcookie('cart','',time()-1);
header("Location:6-28(1).html");
}

?>
```

程序 6-28(3).php 显示购物车页面。

程序 6-28(3).php

```
<?php
session_start();
 include("shopping_cart.php");
 echo "<p align='center'>Session 内容为：  ";
 print_r($_SESSION['cart']);
 echo "</p>" ;
 $cart=new cart();
 $cart_array=$cart->read_session();
 if(count($cart_array)>0)
 {
   $total =0;
   echo '<div align="center"><table border="1" cellpadding="0" cellspacing=
"0"  align="center"  >'
   .' <tr><td colspan="5"><div align="center">购物车你选中的商品如下：
</div></td></tr><tr><td>'
   .'名称</td><td>数量</td><td>单价</td><td>价格</td><td>操作</td></tr>';
   foreach($cart_array as $good)
   {
```

```
        $total+=$good[1]*$good[2];
    echo '<form method="post" action="6-28(2).php" ><tr><td><input '
        .'type="text" name="cargo" value="'.$good[0]
        .'"readonly="readonly" size="15" /></td><td><input '
        .'type="value" name="num" value="'.$good[1]
        .'"size="6" /></td><td>'.$good[2].'</td><td>'
        .$good[1]*$good[2].'</td><td><input type="hidden" '
        .'name="act" value="cancel"><input type="submit" '
        .'value="移除本项" /></td></tr></form>';
        }
    echo '<form method="post" action="6-28(2).php" ><tr><td>合计</td><td>'
        .' </td><td> </td><td>'.$total
        .'</td><td><input type="hidden" name="act" value="cancel_all">'
        .'<input type="submit" value="全部取消"/></td></tr></form>';
        echo '</table></div>';
        }
        echo'<p align="center"><a href="6-28(1).html">'."继续购  "."</a></p>";
    ?>
```

程序运行结果如图 6-30 和图 6-31 所示。



图 6-30 程序 6-28(1).html 的运行结果

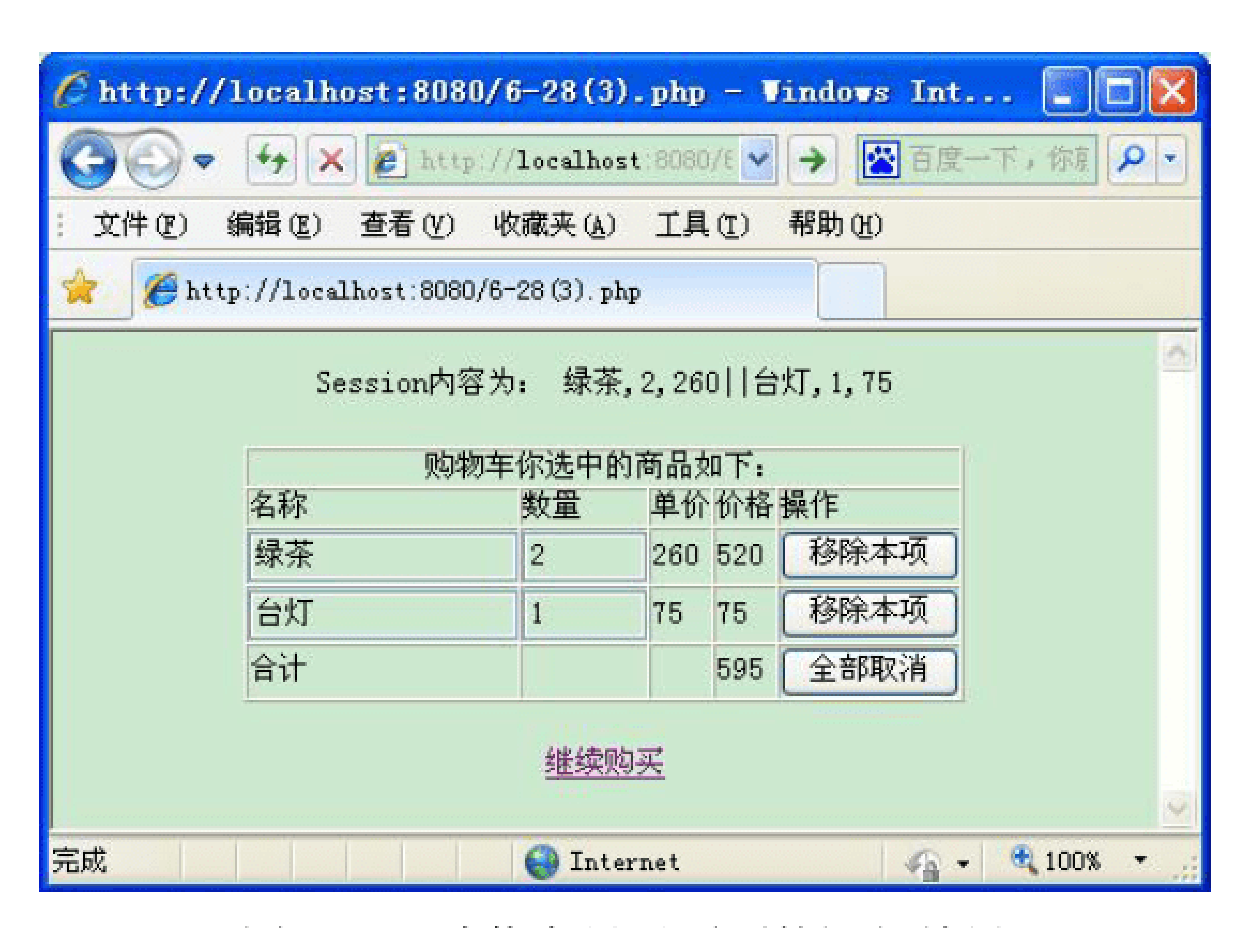


图 6-31 购物车显示页面的运行结果

6.8　本章小结

本章着重介绍了 PHP 的语法基础，主要有预定义常量、自定义常量的定义与使用，变量的定义、变量的类型及相互转换，运算符和表达式，以及程序的流程控制语句。本章还讲解了自定义函数的定义与调用、参数的传递方法以及变量作用域的相关知识。这些知识都是学好 PHP 的基础。本章内容虽然比较多，但难度并不是很大。值得注意的是 PHP 中的自定义函数如果掌握好了，对学习第 7 章的类和对象的相关知识大有好处，这是学好 PHP 面向对象编程的基础。

6.9　练习题

1．PHP 中如何引入外部文件？
2．PHP 中如何定义自定义常量？
3．PHP 中如何进行变量类型的转换？
4．PHP 中如何使用引用变量？
5．PHP 中的操作符有哪些类型？
6．PHP 中实现分支结构和循环结构都有哪些语句？
7．PHP 中自定义函数如何定义，如何使用？
8．PHP 中自定义函数的参数传递有几种方式？
9．PHP 5 的类中变量有几种类型？
10．分别用循环语句和函数实现杨辉三角形。
11．什么是 Cookie？如何使用 Cookie？
12．什么是 Session？Session 主要用在什么方面？

第7章 PHP 数据处理

本章重点

- 数组的处理;
- 函数的运用;
- 字符串操作;
- 图形和图像;
- 时间和日期;
- 正则表达式;
- 文件的处理。

PHP 主要通过数组、函数、字符串、正则表达式、目录与文件、图形图像、时间日期等的操作，实现对数据内容的处理。

7.1 数组

7.1.1 数组的创建

在 C++、Java 等语言中，数组是同类型变量的集合。PHP 的数组中的元素可以为多种类型，操作更灵活，功能更强大。借助于数组的各种特性，可以方便快捷地实现程序的各种功能，学会处理数组能让我们的开发得心应手。

要使用一个数组，必须先创建并命名它。PHP 提供了一个 array 函数来定义一个数组。使用函数 array 来创建数组的语法如下：

```
array( [ key => ] value,…)
```

其中，key 可以是 integer 或 string，是以后存取的标志。当为 integer 时，没有序号意义，value 可以是任何值。

用 array 函数创建数组的方法如程序 7-1.php 所示。

程序 7-1.php

```
<!--程序 7-1.php：用 array 函数创建 PHP 数组-->
```

```
 <?php
    $arr=array(
    0=>6,
    2=>6.666e2,
    1=>"我爱 PHP",
    "str"=>"string",
    );
for ($i=0;$i<count($arr);$i++)
        {
          $print=each($arr);
              echo "$print[value]<br>";
         }
    ?>
```

程序中使用 for 循环用来输出整个数组。其中函数 count 用来计算数组元素的个数，函数 each 返回当前数组指针的索引/值对。程序 7-1.php 的运行结果如图 7-1 所示。



图 7-1　程序 7-1.php 的运行结果

可以采用给数组元素逐个赋值的方法，如程序 7-2.php 所示。

程序 7-2.php

```
<!--程序 7-2.php：逐一给数组元素赋值-->
<?php
 $arr[0]=6;
 $arr[2]=6.666e2;
 $arr[1]="我爱 PHP";
 $arr["str"]="string";
 for ($i=0;$i<count($arr);$i++)
   {
     $print=each($arr);
     echo "$print[value]<br>";
    }
?>
```

其运行结果与程序 7-1.php 相同。

当然，还可以采用下面的更为简洁的方法赋值。

程序 7-3.php

```
<!--程序 7-3.php：数组元素简洁赋值-->
<?php
$arr=array(6,6.666e2,"我爱 PHP","string");
 for ($i=0;$i<4;$i++)
  {
   echo $arr[$i]."<br>";
   }
 ?>
```

使用上面这种简洁方式给数组赋值时，数组的默认下标为 0、1、2、3。程序 7-3.php 的运行结果与程序 7-1.php 的运行结果相同。

PHP 中多维数组与一维数组的区别在于多维数组有两个或多个下标，其用法基本是一样的。程序 7-4 是采用逐一给二维数组元素赋值的方法来创建和使用二维数组的。

程序 7-4.php

```
!--程序 7-4.php：多维数组的逐一赋值法-->
<?php
$arr[0][0]=6;
$arr[0][1]=6.666e2;
$arr[1][0]= "我爱 PHP";
$arr[1]["str"]="string";
for ($i =0;$i<count($arr);$i++)
  {for ($j = 0;$j<count($arr[$i]);$j++)
    {
      $print=each($arr[$i]);
      echo "$print[value]<br>";
     }
  }
?>
```

其运行结果与程序 7-1.php 运行结果相同。

可以使用层次更明显、更容易理解和接受的多维数组赋值方式：嵌套的 array 函数方式创建 PHP 数组并给数组元素赋值，如程序 7-5 所示。

程序 7-5.php

```
<!--程序 7-5.php：用嵌套的 array()函数创建 PHP 数组-->
<?php
 $arr=array
  (
   0=>array
    (
       0=>6,
       2=>6.666e2,
    ),                 //此处应该是“,”，而不是“;”
    1=>array
    (
       0=>"我爱 PHP",
       "str"=>"string"
    )
   );
  for ($i =0;$i<count($arr);$i++)
  {
    for ($j = 0;$j<count($arr[$i]);$j++)
    {
       $print=each($arr[$i]);
       echo "$print[value]<br>";
    }
  }
?>
```

其运行结果与程序 7-1.php 运行结果相同。

7.1.2　数组的操作

数组创建之后，就可以对数组进行遍历、修改、排序等各种操作。PHP 中为用户提供了一系列用来操作数组的函数，这些函数为标准函数，可以直接使用。表 7-1 所示为 PHP 5 提供的常用数组函数。

表 7-1　PHP 5 提供的常用数组函数

函 数 名	功　能
array_change_key_case	返回字符串键名全为小写或大写的数组
array_chunk	将一个数组分割成多个
array_combine	创建一个数组，用一个数组的值作为其键名，另一个数组的值作为其值
array_count_values	统计数组中所有的值出现的次数
array_fill	用给定的值填充数组
array_flip	交换数组中的键和值
array_keys	返回数组中所有的键名
array_map	将回调函数作用到给定数组的单元上
array_merge_recursive	递归地合并一个或多个数组
array_merge	合并一个或多个数组
array_multisort	对多个数组或多维数组进行排序
array_pad	用值将数组填补到指定长度
array_pop	将数组最后一个单元弹出（出栈）
array_product	计算数组中所有值的乘积
array_push	将一个或多个单元压入数组的末尾（入栈）
array_rand	从数组中随机取出一个或多个单元
array_reverse	返回一个单元顺序相反的数组
array_shift	将数组开头的单元移出数组
array_slice	从数组中取出一段
array_splice	把数组中的一部分去掉并用其他值取代
array_sum	计算数组中所有值的和
array_unique	移除数组中重复的值
array_unshift	在数组开头插入一个或多个单元
array_values	返回数组中所有的值
array_walk_recursive	对数组中的每个成员递归地应用用户函数
array_walk	对数组中的每个成员应用用户函数
array	新建一个数组

续表

函　数　名	功　　能
arsort	对数组进行逆向排序并保持索引关系
asort	对数组进行排序并保持索引关系
compact	建立一个数组，包括变量名和它们的值
count	计算数组中的单元数目或对象中的属性个数
current	返回数组中的当前单元
each	返回数组中当前的键/值对并将数组指针向前移动一步
end	将数组的内部指针指向最后一个单元
extract	从数组中将变量导入到当前的符号表
in_array	检查数组中是否存在某个值
key	从关联数组中取得键名
krsort	对数组按照键名逆向排序
ksort	对数组按照键名排序
list	把数组中的值赋给一些变量
natcasesort	用“自然排序”算法对数组进行不区分大小写字母的排序
natsort	用“自然排序”算法对数组排序
next	将数组中的内部指针向前移动一位
prev	将数组的内部指针倒退一位
range	建立一个包含指定范围单元的数组
reset	将数组的内部指针指向第一个单元
rsort	对数组逆向排序
shuffle	将数组打乱
sizeof	count()的别名
sort	对数组排序
uasort	使用用户自定义的比较函数对数组中的值进行排序并保持索引关联
uksort	使用用户自定义的比较函数对数组中的键名进行排序
usort	使用用户自定义的比较函数对数组中的值进行排序

可以看到，PHP 开发者竟然提供了如此丰富的函数。实际上，PHP 提供的数组操作函数多达 110 多个，表 7-1 中并未列出全部函数，而只列出了其中较为常用的函数。

即使是表中列出的函数，这里也不可能一一讲解其使用方法。下面将着重讲解其中最为常用的函数，其他函数读者可以参考 PHP 手册学习其使用方法。有些读者可能会被这密密麻麻的函数吓倒，认为学习 PHP 很难。实际上，每一个学习者都不可能把这所有的函数都记住。除了少量的极为常用的函数需要记住以外，其他大多数函数都没有必要去死记硬背。一种比较好的学习方法是将所有函数浏览一遍，并大体记住其功能。等到编程中遇到

类似问题时，可以通过查找函数手册找到函数的使用方法，然后应用到程序中。实际上很多编程语言的函数库、类库都很庞大，不可能短时间内全部掌握，都有一个逐渐熟悉、积累的过程。

下面以几个函数为例来说明数组处理函数的使用。

1. array 函数

array 函数用来建立一个新数组。函数的参数可以是一个混合类型。下面看一个例子。

程序 7-6.php

```
<!--程序 7-6.php：用 Array 函数建立数组-->
<?php
  $arr1=array(0,1,2,3,4);
  $arr2=array("a"=>0,"b"=>1,"c"=>2,"d"=>3,"e"=>4);
  echo "\$arr1[0]=".$arr1[0];
  echo "<br>";
  echo "\$arr2[\"a\"]=".$arr2["a"];
?>
```

在程序 7-6.php 中，首先用 array 函数定义了拥有 5 个元素的数组$arr1，并且每个元素分别赋值 0，1，2，3，4。然后定义了同样 5 个元素的数组$arr2，并分别赋值 0，1，2，3，4。两个数组的不同是，第一个数组用了默认的数字作为下标；第二个数组用了自定义的字符作为下标。因此最后输出数组元素时也使用了各自对应的下标。程序的运行结果如图 7-2 所示。

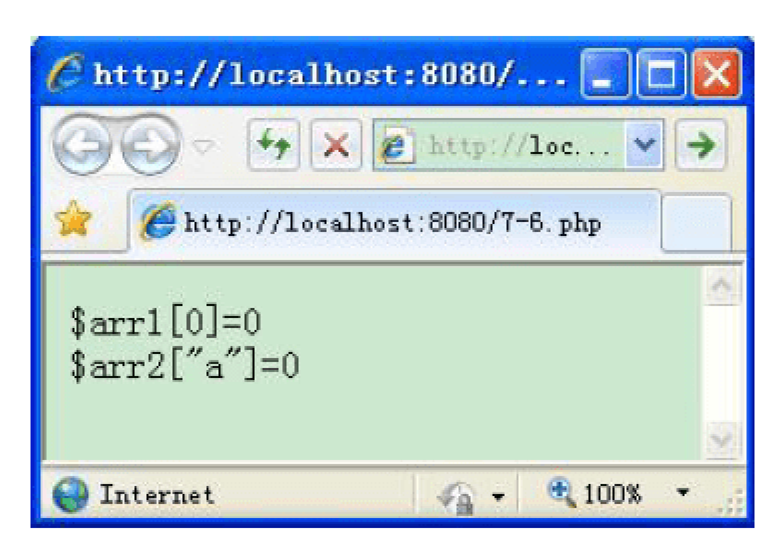


图 7-2　程序 7-6.php 的运行结果

2. count 函数

count 函数用来统计一个数组中元素的个数，在循环遍历一个未知长度的数组时非常有用。看下面的例子。

程序 7-7.php

```
<!--程序 7-7.php：count 函数的使用-->
<?php
  $arr1=array(0,1,2,3,4);
  echo "数组\$arr1 中元素的个数为：".count($arr1);
?>
```

程序运行后将输出：“数组$arr1 中元素个数为：5”。

3. each 函数

each 函数可以返回一个数组中当前元素的键和值，并将数组指针向前移动一步，常常被用在循环中来遍历一个数组。

程序 7-8.php

```
<!--程序 7-8.php：each 函数的使用-->
<?php
  $arr = array("name"=>"Bob","age"=>20,"sex"=>"male","postcode"=>
"100000");
  for($i=0;$i<count($arr);$i++){
  $keyAndValue=Each($arr);
```

```
    echo $keyAndValue["key"]."=>".$keyAndValue["value"];
    echo "<br>";
  }
  ?>
```

在程序 7-8.php 中，首先定义了一个数组$arr，并且为其赋值。值得注意的是，数组下标不是按顺序递增的数字，而是毫无规律的字符串。所以，不能直接用一个递增的数字作为下标输出，循环输出遇到了困难。但是使用 each 函数可以获得这个数组的下标以及下标对应的值，因此就可以使用循环输出每一个元素的下标和值。函数 each($arr)将$arr 数组中当前元素的下标和值都存放到另外一个数组$kav 中，然后将数组指针指到下一个元素。$kav 数组的下标分别为 key 和 value。这样，只需要调用$kav["key"]和$kav["value"]即可获得该元素的下标和值。输出这两个值后，本次循环结束，执行下一次循环，这样又输出了下一个元素的值。依此类推，整个数组都被动态循环输出了。程序的运行结果如图 7-3 所示。

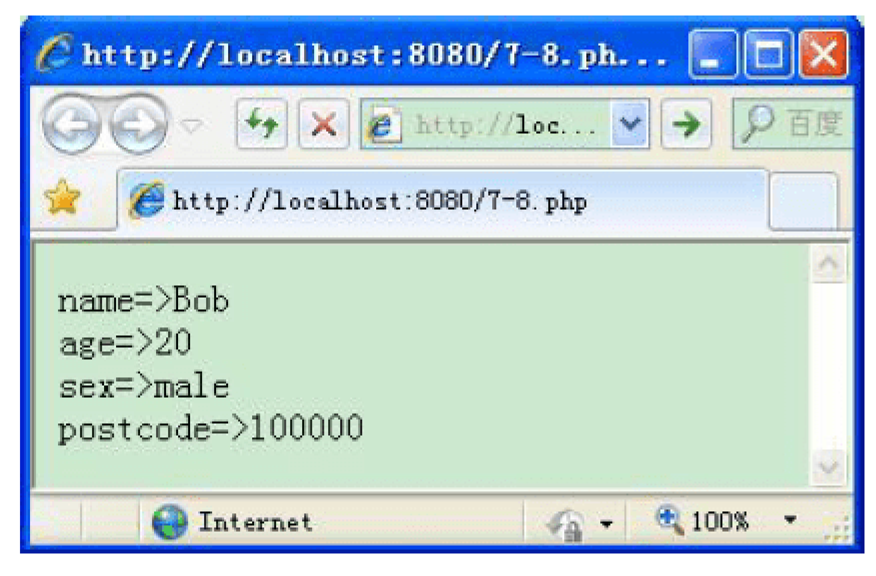


图 7-3　程序 7-8.php 的运行结果

通过程序 7-8.php 看到了 each 函数的妙用。其程序 7-7.php 中的代码还可以继续简化，不用 count 函数统计数组元素的个数，也可以实现动态循环输出一个未知长度的数组。

程序 7-9.php

```
<!--程序 7-9.php：each 函数的使用-->
<?php
$arr = array("name"=>"Bob","age"=>20,"sex"=>"male","postcode"=>"100000");
  while($kav=each($arr)){
  echo $kav["key"]."=>".$kav["value"];
  echo "<br>";
  }
?>
```

程序 7-9.php 的代码比程序 7-8.php 简洁，实现的效果却完全相同。程序 7-9 利用了 each 函数一个重要性质，那就是当数组已经到达末尾时 each 函数返回 false。通过前面所学的知识，读者知道 false 是一个布尔值，表示“否”。因此它正好可以作为 while 循环的结束条件。这样，可以用一个 while 循环每次读取$arr 数组中的一个元素，不管数组有多少个元素，当指针到达末尾时，each($arr)返回 false，循环结束，程序执行完成。同样实现了动态输出未知长度数组的功能。程序的运行结果如图 7-3 所示。

通过程序 7-9.php 也可以说明，有时实现同一个功能，可以选择多种途径。作为程序开发人员，应该尽量选择更加简洁、高效的途径。

4. current 函数、reset 函数、next 函数和 prev 函数

之所以要将这 5 个函数并列起来介绍，是因为这 5 个函数的作用相似——它们都用来操作数组内部的指针。在 PHP 中，使用一个内部指针来指向一个数组。需要访问数组中的某一元素时，只需要将指针移动到该元素的位置，即可取出该元素，这大大方便了用户对数组的操作。下面先详细说明这 5 个函数的作用，然后通过一个例子验证其使用效果。

current()：返回当前内部指针所指的元素的值。当到达数组末尾时返回 false。

reset()：将内部指针指向数组的第一个元素，并返回其值。数组为空时返回 false。

end()：将内部指针指向数组的最后一个元素，并返回其值。

next()：将数组指针指向当前元素的下一个元素，并返回其值。到达末尾时返回 false。

prev()：将数组指针指向当前元素的上一个元素，并返回其值，当到达顶端时返回 false。

上面 5 个函数的返回值均为 mixed 类型，根据数组元素值的类型不同而返回不同的类型。在这里要注意 current 函数和 next 函数的不同。它们虽然都是取出一个元素值，但是 current()并不移动指针，也就是说 current()返回的是未移动指针之前所指向的元素的值，而 next()返回的是移动指针之后所指向的元素的值。

程序 7-10.php

```
<!--程序 7-10.php：数组内部指针移动-->
<?php
  $arr=array(1,2,3,4,5,6,7,8,9,10);
  echo "调用 current():".current($arr);
  echo "<br>";
  echo "再次调用 current():".current($arr);
  echo "<br>";
  echo "调用 next():".next($arr);
  echo "<br>";
  echo "调用 reset():".reset($arr);
  echo "<br>";
  echo "调用 end():".end($arr);
  echo "<br>";
  echo "调用 prev():".prev($arr);
?>
```

程序 7-10.php 中定义了一个数组$arr，并且用 10 个数字对其进行了初始化。然后分别调用上述 5 个函数来观察其运行效果。为了使输出结果直观，在每一次调用之后都输出一个换行。程序运行结果如图 7-4 所示。

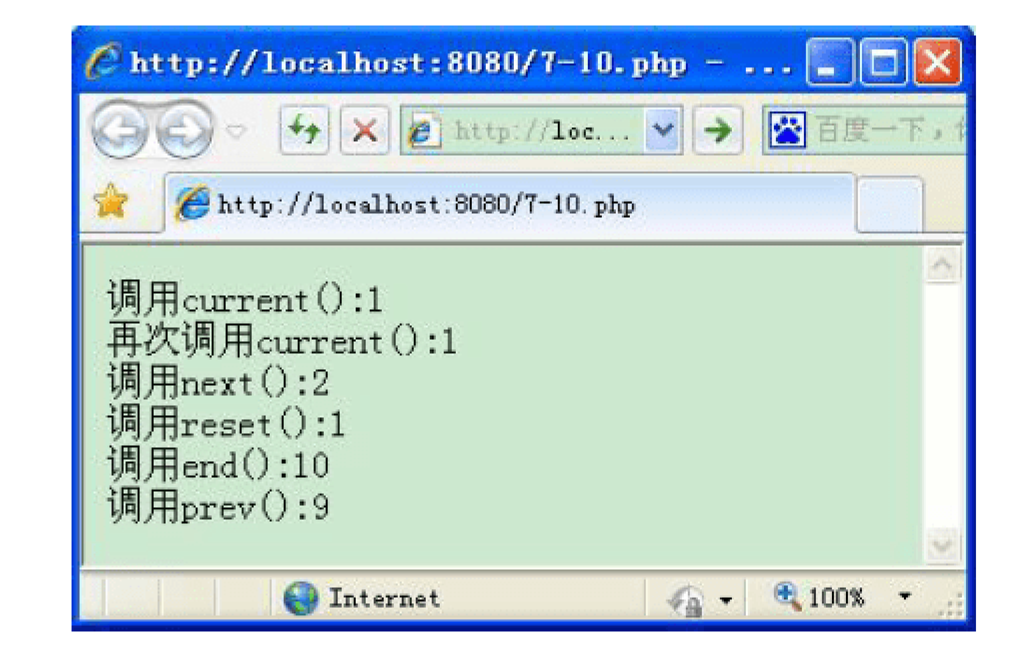


图 7-4　程序 7-10.php 的运行结果

下面来分析程序的运行流程和对应的输出结果：

（1）数组初始化完成，内部指针指向第一个元素（元素值为 1）。

（2）第一次调用 current 函数，返回当前元素值 1，指针不变。

（3）再次调用 current 函数，由于内部指针不变，仍然返回 1。

（4）调用 next 函数，内部指针指向下一个元素，并返回其值（返回 2）。

（5）调用 reset 函数，内部指针再次指向第一个元素，返回 1。

（6）调用 end 函数，内部指针指向最后一个元素，并返回其值（返回 10）。

（7）调用 prev 函数，内部指针指向前一个元素，并返回其值（返回 9）。

关于 PHP 的数组函数，就介绍到这里。表 6-1 中列出的其他函数，如果读者感兴趣可以自行编写程序进行测试。函数的参数、返回值类型等均可以通过查看 PHP 手册获得。

7.2 函数

函数是一个给出了调用接口的自包含的独立代码块。函数常用来实现某一功能，执行特定任务并返回执行结果。函数具有控制程序设计的复杂性，提高软件的重用性、可靠性和易维护性等优点，有利于提高软件开发效率。PHP 中有 3 种函数类型，即自定义函数、内置函数和变量函数。

7.2.1 自定义函数

用户自定义的函数称自定义函数，PHP 中允许用户定义自己的函数。定义函数的语法为：

```
function 函数名(形式参数列表)
{
          函数体;
          return 返回值;
}
```

PHP 中的函数可以有返回值，也可以没有返回值。在函数的名称上，PHP 对于大小写的管理比较松散。可以在定义函数时写成大写的名字，而在使用时使用小写的名字。不过，PHP 对用户自定义函数的函数名还是有一些具体的要求：

① 不能与 PHP 的内部函数同名。

② 不能与 PHP 的关键字重名。

③ 不能以数字或下划线开头。

④ 不能包含点号“.”和中文字符。

函数体是实现函数功能的语句集合。函数体中即使只有一条语句，外面的大括号也不能省略。

函数调用的语法为：

```
函数名（实际参数列表）;
```

实际参数列表要与形式参数列表相对应（有默认参数时，实际参数还要与默认参数对应）。如果实际参数比形式参数多，多余的参数会被自动舍弃；如果实际参数比形式参数少，实际参数会被一一填入形式参数中，不足部分以空参数代替。实际参数和形式参数之间的传递机制会在 7.2.2 节中详细介绍。

如果函数有返回值，还可以利用函数调用为变量赋值，其语法为：

```
变量=函数名（实际参数列表）;
```

程序 7-11.php 演示了函数的定义和调用。

程序 7-11.php

```
<!--程序 7-11.php：函数的定义和调用=>求阶乘-->
<?php
    function factorial($n)
    {
        $result=1;
```

```
        for ($i=2;$i<=$n;$i++)
            $result*= $i;
        return $result;
    }
    for ($i=1;$i<=6;$i++)
    {
        $num=factorial($i);
        echo $i."!=".$num."<br>";
    }
?>
```

程序 7-11.php 的运行结果如图 7-5 所示。

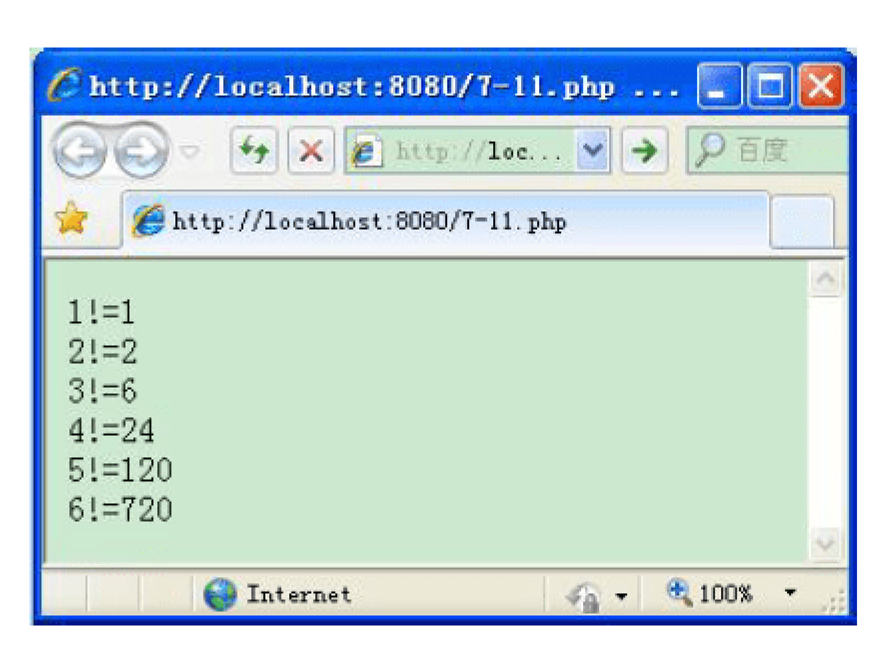


图 7-5 程序 7-11.php 的运行结果

7.2.2 参数传递

在调用函数时，要填入与函数形式参数个数相同的实际参数（有默认参数的除外），在程序运行过程中，实际参数就会传递给相应的形式参数，然后在函数中实现对数据的处理和返回。实际参数向形式参数传递的过程中，共有值传递、引用传递、默认参数 3 种机制。

1. 值传递

值传递就是将实际参数的值复制到形式参数中，然后使用形式参数在函数内部进行运算，函数调用结束后，实际参数的值不会发生改变。

用这种方式调用的函数一般都有返回值或输出值；否则函数调用没有实际意义。程序 7-11.php 中就是使用的这种传递方式。

2. 引用传递

如果要想形式参数改变时实际参数也发生相应的改变，就要使用引用传递的方式。参数的引用传递也有两种方法：

（1）定义函数时，在形式参数前面加上“&”符号。例如：

```
function fun(&$var1){…}
```

（2）函数调用时，在实际参数前面加上“&”符号。例如：

```
function fun($var1){…}
fun(&$var2)
```

如果形式参数$var1 的值在函数中发生改变，实际参数$var2 的值也会发生相应的改变。

3. 默认参数

PHP 还支持有默认值的参数，即定义函数时可以为一个或多个形式参数指定默认值。程序 7-12.php 演示了函数的参数传递的 3 种不同方式。

程序 7-12.php

```
<!--程序 7-12.php：函数参数的传递-->
<?php
    function myfun1($var1)     //值传递测试函数
    {
        $var1=88;
    }
    function myfun2(&$var1)   //引用传递测试函数
```

```
    {
        $var1=88;
    }
    function myfun4($string,$color="red")     //默认参数传递测试函数
    {
        echo "<font color=".$color.">".$string."</font>";
    }
    $var1=66;
    $string="这是红色字体！";
    echo "初始值:\$var=".$var1."<br>";
    myfun1($var1);
    echo "值传递结束后:\$var=".$var1."<br>";
    myfun2($var1);
    echo "引用传递结束后:\$var=".$var1."<br>";
    echo "默认参数的效果为:";
    @myfun4($string);
?>
```

程序 7-12.php 的运行结果如图 7-6 所示。

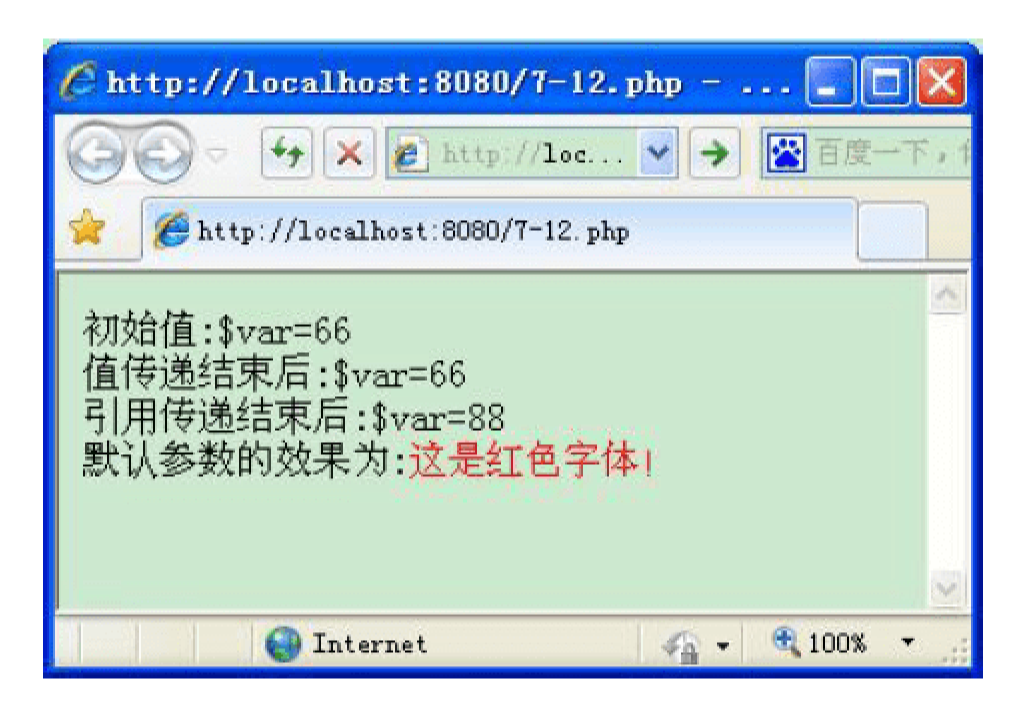


图 7-6　程序 7-12.php 的运行结果

7.2.3　变量函数

PHP 支持变量函数的概念，可以方便地利用变量实现对函数的调用。使用的基本格式为：

```
function fun(){…}
$var="fun";
$var();
```

其中，“$var();”就相当于调用函数 fun，$var()为可变函数。变量函数与普通函数调用时的最大区别就在于可变函数前面有$，有此符号，系统就会认为是变量函数。

程序 7-13.php 为变量函数的示例。

程序 7-13.php

```
<!--程序 7-13.php: 变量函数-->
<?php
  function myfun()
   {
```

```
        return "变量函数执行成功！";
      }
    $var="myfun";
    echo $var();
    echo "<br>";
    $var="myfun()";
    echo $var;
  ?>
```

程序 7-13.php 的运行结果如图 7-7 所示。

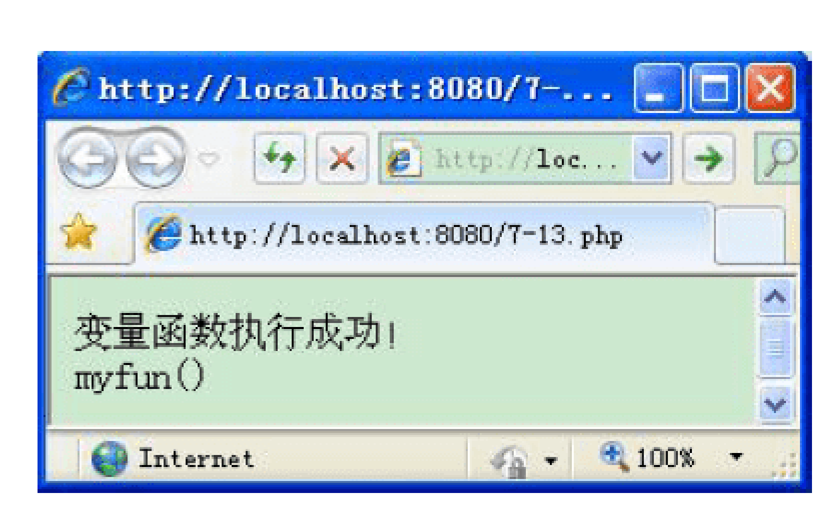


图 7-7　程序 7-13.php 的运行结果

7.2.4　内置函数

内置函数由 PHP 开发者编写并已嵌入到 PHP 当中，用户可以在程序中直接使用。使用 PHP 开发者提供的大量的内置函数可以轻松地完成很多操作。可以说，学习和使用内置函数是学习 PHP 的重要步骤，也是用 PHP 编写复杂程序的重要前提。

PHP 中的内置函数也大体分为两大类，一是标准函数库；二是扩展函数库。标准函数库中的函数存放在 PHP 内核中，可以在程序中直接使用，不需要其他任何声明、载入等操作。扩展函数库中的函数一般不能直接使用，而是按照个人不同的需求有选择地使用。这些扩展库函数按照功能的不同被分门别类地封装在多个 DLL 函数库中，这些 DLL 库函数存放在 PHP 安装文件夹下。在 PHP 5 中，扩展函数被存放在 PHP 安装目录的 ext/子目录下，如图 7-8 所示。

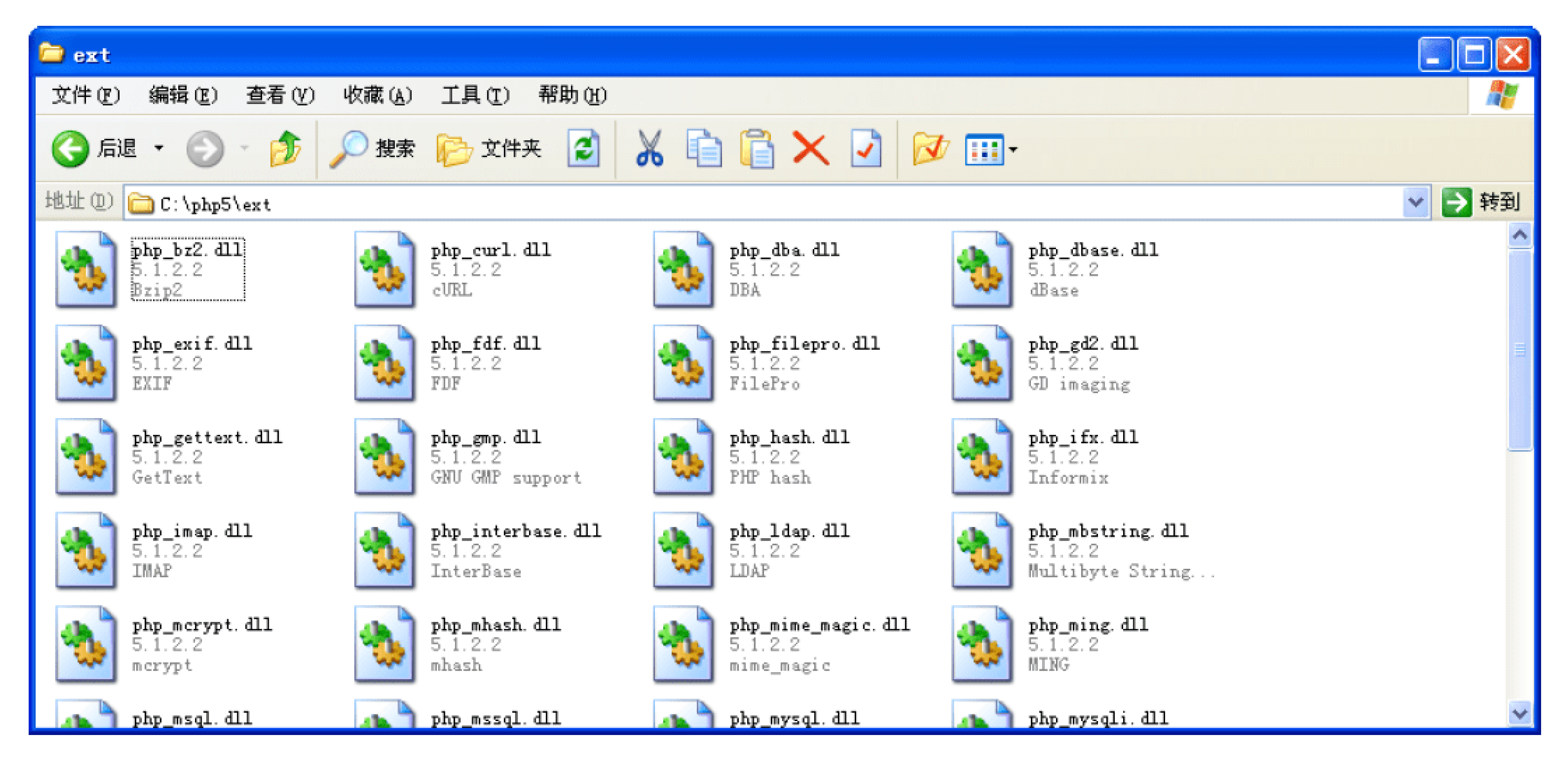


图 7-8　PHP 中的内置扩展函数库

当用户需要用到扩展函数库中的这些函数时，只需在 php.ini 配置文件中将此扩展库打开即可，它们在 php.ini 中的位置如图 7-9 所示。

打开一个扩展库的方法很简单，只需要将“;extension=php_xxx.dll”前面的分号“;”去掉，并保存 php.ini 文件，然后重新启动 IIS 或 APACHE，此时 php.ini 生效，此扩展随即可以使用。

在本书的程序中用到的函数，绝大多数都属于标准内置函数，不需要对 PHP 做特殊的配置就能够直接使用。如果用到扩展函数库中的函数，会进行说明。读者如果在编写程序过程中，遇到有的 PHP 函数不能使用的情况，应当考虑是否因为没有打开相应的扩展库。如果没有打开相应的扩展库，PHP 系统一般会给出“Fatal error: Call to undefined function…”的提示。

```
php.ini - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

;;;;;;;;;;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;;;;;;;;;;

;extension=php_mbstring.dll
;extension=php_bz2.dll
;extension=php_curl.dll
;extension=php_dba.dll
;extension=php_dbase.dll
;extension=php_exif.dll
;extension=php_fdf.dll
;extension=php_filepro.dll
;extension=php_gd2.dll
;extension=php_gettext.dll
;extension=php_ifx.dll
;extension=php_imap.dll
;extension=php_interbase.dll
;extension=php_ldap.dll
;extension=php_mcrypt.dll
;extension=php_mhash.dll
;extension=php_mime_magic.dll
;extension=php_ming.dll
;extension=php_mssql.dll
;extension=php_msql.dll
;extension=php_mysql.dll
;extension=php_oci8.dll
;extension=php_openssl.dll
;extension=php_oracle.dll
;extension=php_pgsql.dll
```

图 7-9　扩展函数库在 php.ini 中的位置

7.3 字符串

字符串是 PHP 程序相当重要的一部分操作内容，程序传递给人们的可视化信息，绝大多数都是靠字符串实现的。美国国家标准学会（American National Standard Institute，ANSI）制定了一整套可见的字符（如大写字母 A）和不可见的字符（如回车符、退格符等控制字符），共 127 个字符。由零个或者多个字符组成的有限序列，也就是 n（n≥0）个字符的集合，叫做字符串。在书写上，字符串一般都写在成对的单引号或者双引号中。例如：

```
$text_a = "This is a test message";//一个普通的字符串，用双引号
$text_b = '1234567890';            //一个全由数字组成的字符串
$text_c = chr(10);                 //用一个不可显示的 ASCII 字符组成的字符串
$text_d = '';                      //一个空字符串
$text_e = "Hello!',Jack said."     //双引号中嵌套使用成对的单引号
$text_f = '<a href="http: //news.my.com" target="_blank">我的新闻网</a>';
//是一个由 XHTML 代码组成的字符串，单引号中嵌套成对使用的双引号
```

7.3.1　字符串处理函数

在 Web 编程中，字符串是使用最为频繁的数据类型之一。PHP 不是一门强类型语言，因此很多数据都可以方便地作为字符串处理。字符串操作是编程中极为常用的操作，从简单的打印输出一行字符串到复杂的正则表达式等，处理目标都是字符串。PHP 提供了大量实用的函数，可以完成许多复杂的字符串处理工作。PHP 提供的字符串处理函数及其功能如表 7-2 所示。

表 7-2　PHP 提供的字符串处理函数

函　数　名	功　　能
addcslashes	像 C 一样使用反斜线转义字符串中的字符
addslashes	使用反斜线引用字符串
bin2hex	将二进制数据转换成十六进制表示
chop	rtrim()的别名
chr	返回指定的字符
chunk_split	将字符串分割成小块
convert_cyr_string	将字符由一种 Cyrillic 字符转换成另一种
convert_uudecode	对一个未编码字符串进行编码
convert_uuencode	对一个字符串进行解码
count_chars	返回字符串所用字符的信息
crc32	计算一个字符串的 crc32 多项式
echo	输出字符串
explode	使用一个字符串分割另一个字符串
fprintf	将格式化字符串写入流
html_entity_decode	将 HTML 标记转换为特殊字符
htmlspecialchars	将特殊字符转换为 HTML 标记
implode	合并数组元素到一个字符串中
join	Implode 函数的别名
ltrim	去除字符串左侧空格
md5_file	用 md5 算法对文件进行加密
md5	用 md5 算法对字符串进行加密
nl2br	将换行符替换成 HTML 的换行符
number_format	将一个数字格式化成三位一组
ord	返回一个字符的 ASCII 码
print	输出字符串
printf	输出格式字符串

续表

函 数 名	功 能
rtrim	去除字符串右侧空格
sprintf	返回一个格式字符串
str_pad	用一个字符串填充另外一个字符串到一定长度
str_repeat	重复输出一个字符串
str_replace	字符串替换
str_shuffle	随机打乱一个字符串
str_split	将字符串转换成数组
str_word_count	统计字符串中的单词数
strchr	strstr()的别名
strcmp	字符串比较大小
strip_tags	过滤掉字符串中的 PHP 和 HTML 标记
strlen	获得字符串的长度（所占字节数）
strpbrk	以子串中的任意一个字符第一次出现的位置为界将字符串分成两部分，并返回后半部分
strpos	查找一个子串在字符串中第一次出现的位置
strrpos	查找一个子串在字符串中最后一次出现的位置
strrev	将一个字符串顺序倒转
strrchr	查找一个字符在一个字符串中最后一次出现的位置并返回从此位置开始之后的字符串
strstr	查找一个子串在一个字符串中第一次出现的位置，并返回从此位置开始的字符串
strstr	strrchr 函数的别名
strtok	将字符串打碎成小段
strtolower	将字符串中的字符全部变为小写
strtoupper	将字符串中的字符全部变为大写
strtr	批量字符替换
substr_count	统计一个子串在字符串中出现的次数
substr_replace	在字符串内部定制区域内替换文本
substr	截取字符串的一部分
trim	去除字符串首尾的连续空格
ucfirst	将字符串首字母大写
ucwords	将字符串中每个单词的首字母大写

表 7-2 列出了常用的 PHP 字符串操作函数。以下将最为常用的字符串处理函数分别详细介绍。

1. trim()、ltrim()、rtrim()、chop()和 strlen()

这 5 个函数中前 4 个函数的功能类似，因此将其放在一起介绍。chop()与 rtrim()作用相同，都是去除字符串右端的空格。ltrim()用来去除字符串左端的空格，而 trim()用来去除字符串左右两端的空格。

下面来看一个例子，其中用到了另外一个字符串处理函数 strlen()来获得字符串的长度。

程序 7-14.php

```
<!--程序 7-14.php：去除字符串两端空格-->
<?php
    $str="  你看不到我 我是空格 ";
    echo "方括号中为原始字符串：[".$str."]<br>";
    echo "原始字符串长度：".strlen($str)."<br>";
    $str1=ltrim($str);
    echo "执行 ltrim()之后的长度：".strlen($str1)."<br>";
    $str2=rtrim($str);
    echo "执行 rtrim()之后的长度：".strlen($str2)."<br>";
    $str3=trim($str);
    echo "执行 trim()之后的长度".strlen($str3)."<br>";
    echo "去掉首尾空格之后的字符串：[".$str3."]";
?>
```

程序的运行结果如图 7-10 所示。

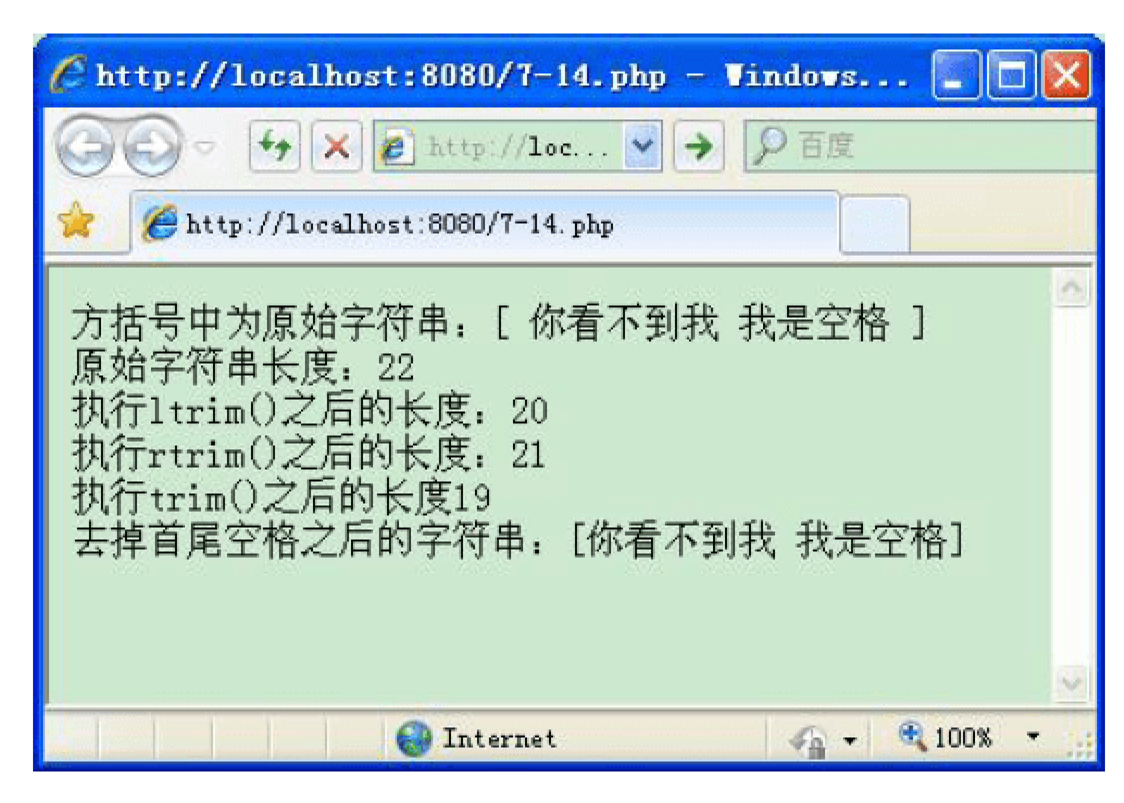


图 7-10　程序 7-14.php 运行结果

在程序 7-14.php 中，首先构造了一个字符串$str，这个字符串由 9 个汉字和 4 个空格组成，4 个空格中有两个在左侧，一个在中间；一个在右侧（由于浏览器会忽略连续的空格，因此在浏览器中连续的两个空格的显示效果与一个空格相同）。由于每个汉字占 2 字节，每个英文半角空格占 1 字节，因此初始字符串的长度为 9×2+4=22。用 strlen()来输出其长度。

首先，执行 ltrim()，将返回结果存放在$str1 中。由于 ltrim()会去掉字符串左侧的所有连续的空格，因此两个空格就被去掉，$str2 的字符串长度为 20。

然后，执行 rtrim()，将返回结果存放在$str2 中。rtrim()去掉了字符串$str 的右侧一个空格，因此$str2 的长度为 21。

最后，执行 trim()。trim()去除字符串左右两侧的所有空格，因此左侧的两个空格和右侧的一个空格被去掉，剩余的部分长度为 19。通过$str3 的输出也可以看出，字符串两侧的空格已经消失。

去除连续的空格往往用在做字符串比较之前。要比较两个字符串是否相同时，如果其中一个字符串首尾带有空格，那比较结果就会为假。如"abc"和"abc"这两个字符串，看似内容完全相同，但由于后者多了一个空格，在比较时会返回 false。因此比较两个字符串变量的值是否相同时，往往首先用 trim 函数处理一下两侧的空格。

值得注意的是，trim 系列函数只去除字符串两侧的空格，而不会去除中间的空格。如程序 7-14.php 中，“你看不到我”和“我是空格”之间有一个空格。调用这 3 个函数之后空格仍然存在，说明字符串中间的空格不会受影响。如果确实想去除掉一个字符串中的所有空格，可以使用后面要讲的字符串替换函数来实现。

2. ucwords()、ucfirst()、strtoupper()、strtolower()和 str_word_count()

这 5 个函数对字符串中的单词进行处理，包括转换大小写转换、计算单词个数等。还是通过程序 7-15.php 了解它们的用法。

程序 7-15.php

```
<!--程序 7-15.php：字符串处理中的单词处理-->
<?php
  $str="ni hao, wo jiao Wang Xiao-ming.";
  echo "原始字符串：".$str."<br>";
  $str1=ucfirst($str);
  echo "执行 ucfirst()之后：".$str1."<br>";
  $str2=ucwords($str);
  echo "执行 ucwords()之后：".$str2."<br>";
  $str3=strtoupper($str);
  echo "执行 strtoupper()之后：".$str3."<br>";
  $str4=strtolower($str);
  echo "执行 strtolower()之后：".$str4."<br>";
  echo "字符串中一共有：".str_word_count($str)."个单词。";
?>
```

程序的运行结果如图 7-11 所示。

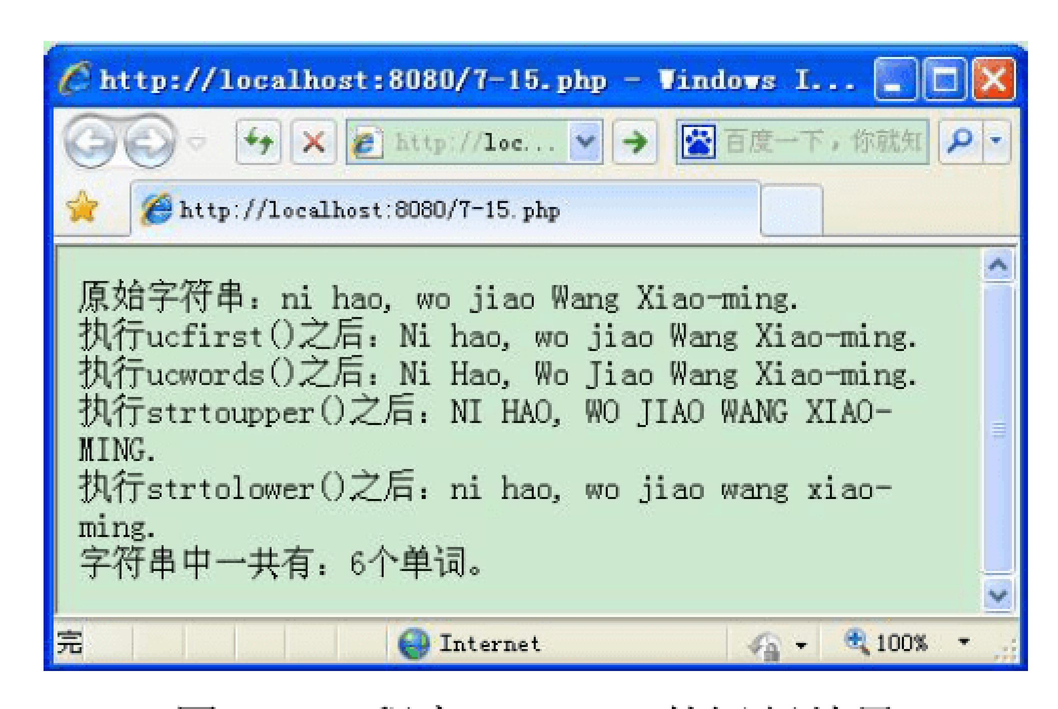


图 7-11 程序 7-15.php 的运行结果

程序 7-15.php 中，构造了一个包含有 6 个单词、大小写混合的字符串，并用它来测试函数的运行结果。程序调用 ucfirst 函数将整个字符串首字母变为大写，调用 ucwords 函数将每个单词的首字母变为大写，调用 strtoupper 函数将全部字母都变成大写，调用 strtolower 函数将所有字母变成小写，最后调用 str_word_count 函数统计字符串中的单词个数。

7.3.2　字符串查找函数

程序中经常用到在一个字符串中查找某个字符或者某个子串的操作；对字符串中的某些字符按照用户的需求进行替换操作以及截取字符串的一部分等。PHP 都已经准备了一系列函数实现这些操作，用户只需要了解函数的使用方法，即可轻松实现。

常用的字符串查找函数有 substr_count()、strpos()、strrpos()、strstr()、strrchr()等。它们的使用方法和功能如下。

1. substr_count 函数

substr_count 函数的格式如下：

```
int substr_count ( string haystack, string needle [, int offset [, int length]] )
```

substr_count 函数用来统计一个字符串 needle 在另一个字符串 haystack 中出现的次数。该函数返回值是一个整数。有两个可选参数：offset 和 length，分别表示要查找的起点和长度。值得注意的是，offset 是从 0 开始计算，而不是从 1 开始计算的。

程序 7-16.php

```
<!--程序 7-16.php：用 substr_count 函数统计字符串出现次数-->
<?php
 $str="I am an abstract about abroad.";
 echo substr_count($str,"ab");
 echo ", ";
 echo substr_count($str,"ab",6,4);
?>
```

本例的输出结果为“3，1”。

substr_count($str,"ab")的作用是返回字符串"ab"在字符串$str 中的次数，由于"ab"在整个字符串中出现了 3 次，因此值为 3。

substr_count($str,"ab",6,4)的作用是返回字符串"ab"在$str 中的从第 6 个字符开始（包含第 6 个字符，而且从 0 开始计算），往后数 4 个字符为止（即第 9 个字符）之间的字符串中出现的次数。这个描述看起来非常拗口和难懂。不妨换一种描述方法：参数“6，4”限定了查找字符串的范围。不指定参数时，substr_count 函数从整个字符串$str 中查找"ab"的出现次数，而指定了参数之后，substr_count 函数从指定的范围内查找"ab"的出现次数。这个范围就是从字符串的第 6 个字符开始到第 9 个字符为止的 4 个字符。具体到本例中，就是"n ab"（注意，n 和 a 之间的一个空格也算一个字符）。显然，在这个范围内，"ab"只出现了一次，因此返回 1。

2. strrpos 函数和 strpos 函数

strrpos 函数的格式如下：

```
int strrpos ( string haystack, mixed needle [, int offset] )
```

该函数返回字符 needle 在字符串 haystack 中最后一次出现的位置。这里 needle 只能是一个字符，而不能是一个字符串。如果提供一个字符串，PHP 也只会取字符串的第一个字符，其他字符无效。参数 offset 也是用来限制查找的范围。

strrpos 函数的格式如下：

```
int strpos ( string haystack, mixed needle [, int offset] )
```

该函数与 strrpos 函数仅一字之差，但功能相差很大。strpos 函数中的 needle 参数允许使用一个字符串，而且返回的是这个字符串在 haystack 中第一次出现的位置，而不是最后一次出现的位置。

程序 7-17.php

```
<!--程序 7-17.php：字符串查找函数的使用(一)-->
<?php
    $str="I am an abstract about abroad.";
    echo "原始字符串为：".$str."<br>";
    echo "ab 在字符串中的第一次出现位置为：".strpos($str,"ab")."<br>";
    echo "ab 在字符串中的最后一次出现位置为：".strrpos($str,"ab")."<br>";
    echo "abcd 在字符串中第一次出现的位置为：".strpos($str,"abcd");
?>
```

本程序的运行结果如图 7-12 所示。

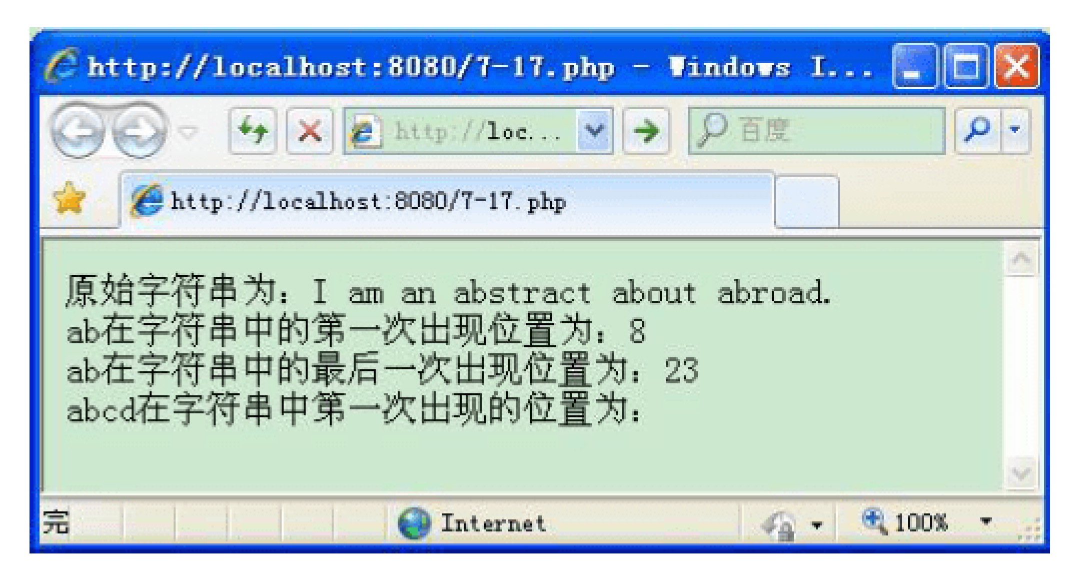


图 7-12 程序 7-17.php 的运行结果

在程序 7-17.php 中，首先构造了一个包含多个"ab"的字符串。然后，分别调用 strpos 和 strrpos 函数来获得"ab"子串在字符串中第一次和最后一次出现的位置。输出结果为 8 和 23。这里有两点值得注意：第一点是这里的 8 和 23 都是指从 0 开始计算的，而且是从子串的第一个字母出现的位置开始计算。例如，子串为"ab"，找到"ab"之后，以 a 的位置序号作为函数的返回值，而不是 b 的位置序号。第二点是如果要查找的字符串不存在，则返回布尔值 false。由于 false 无法直接输出，因此最后查找"abcd"子串时没有任何输出。

3. strstr 函数和 strrchr 函数

strstr 函数和 strrchr 两个函数的格式分别是：

```
string strstr ( string haystack, string needle )
string strrchr ( string haystack, string needle )
```

由此可见，这两个函数均返回一个字符串，而不是返回一个表示位置的整数。两个函数函数名不同，使用方法完全相同，但是其作用略有不同。strstr 函数用来查找一个子串

needle 在字符串 haystack 中第一次出现的位置，并返回从此位置开始的字符串。strrchr 函数查找一个字符 needle 在字符串 haystack 中最后一次出现的位置并返回从此位置开始之后的字符串。

程序 7-18.php

```
<!--程序 7-18.php：字符串查找函数的使用(二)-->
<?php
    $str="千山鸟飞绝，万径人踪灭，孤舟蓑笠翁，独钓寒江雪。";
    echo "1.原始字符串为：".$str."<br>";
    echo "用 strstr 函数搜索"，"的返回结果：".strstr($str,"，")."<br>";
    echo "用 strstr 函数搜索"孤舟"的返回结果：".strstr($str,"孤舟")."<br>";
    $str2="I have a great dream.";
    echo "2.原始字符串为：".$str2."<br>";
    echo "用 strrchr 函数搜索"e"的返回结果：".strrchr($str2,"e")."<br>";
    echo "试图用 strrchr 函数搜索"at"的返回结果：".strrchr($str2,"at");
?>
```

程序的运行结果如图 7-13 所示。

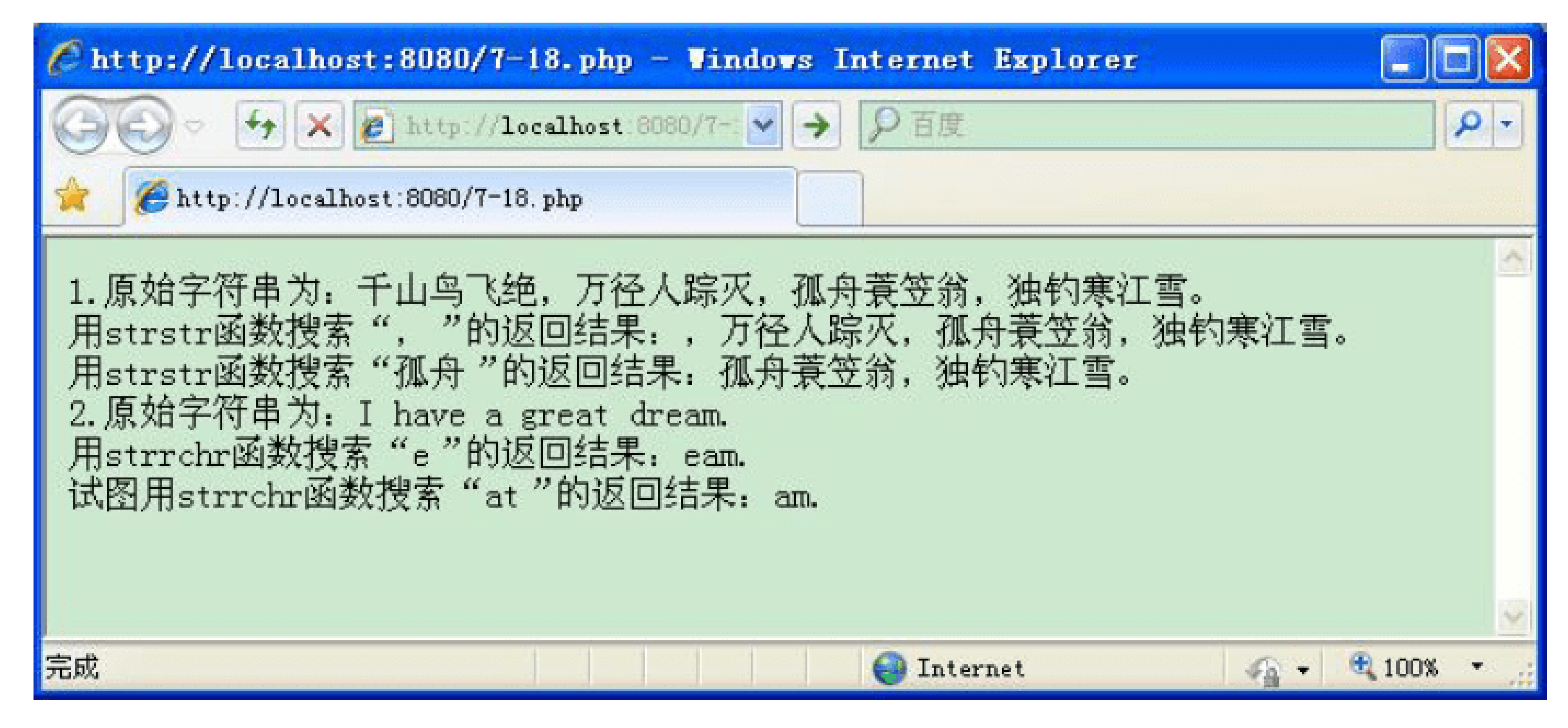


图 7-13　程序 7-18.php 的运行结果

通过深入分析本例的输出结果，就能够准确地把握 strstr 和 strrchr 函数的功能特点。

首先，在第一个字符串中，用 strstr 函数搜索逗号“，”。该函数返回字符串中第一次出现“，”的位置之后的字符串。由于第一次出现逗号是在"千山鸟飞绝"的“绝”字之后，函数的返回结果就是“，万径人踪灭……”（注意逗号本身也会被返回）。为了证明 strstr 函数可以使用一个字符串而不仅仅单个字符作为参数，又在字符串中搜索“孤舟”，显然应当返回“孤舟蓑笠翁……”。和程序的运行结果相同。

然后，又构造了一个英文字母构成的字符串“I have a great dream.”。用 strrchr 函数在字符串中查找 e，返回字符串中最后一次出现 e 位置之后的内容，程序中 3 次出现 e，但最后一次出现是在 dream 中，于是函数返回“eam.”（e 本身也被返回）。最后测试是否可以把一个字符串作为参数传递给 strrchr 函数，在字符串中查找字符串“at”。如果该函数支持字符串参数，按照上面的分析，应当返回“at dream.”。但是根据图 7-13 的运行结果可知，返回的却是“am.”。为什么呢？因为 strrchr 函数不支持字符串参数。如果提供了字符串参数，会自动截取字符串的第一个字符作为参数。也就是说参数 at 和参数 a 所起的作用一样。于是函数返回字符串中最后一次出现 a 之后的内容，也就是“am.”。

可能有读者会问，为什么要构造一个英文的字符串来讲解 strrchr 函数呢？通过刚才的分析就已经能够得到答案。因为每一个汉字都占两个字节，在函数中两个字节会被认为是多个字符（英文中一个字符占一个字节）。因此，strrchr 函数就无法支持中文，也就是说不能把一个或多个中文字符作为参数传递给 strrchr 函数。

除了 strrchr 函数之外，PHP 中还有很多函数无法直接处理中文，这里不一一列出，读者在学习 PHP 和编写程序时应当多加注意。

7.3.3 字符串替换函数

字符串替换是 Web 编程当中极为常用的操作，如要过滤用户提交的不文明的词语，或处理掉字符串中包含的危险脚本，替换掉某些关键词等。PHP 提供了一些函数完成字符串替换操作，如 nl2br()、str_replace()等。

1. nl2br 函数

该函数的名字看起来比较怪，中间包含一个数字 2，用汉语念起来似乎有点别扭。实际上这里的 2 在英文中念 two，与 to 谐音。这里的 2 实际上就是 to 的一种缩写。明白了这一点之后，函数名字和功能就一目了然了。在很多中文参考资料中，将此函数的功能描述为“将换行符替换成 XHTML 的换行符
”，本书也沿用这一解释。但是如果查阅英文版 PHP 手册，会发现大意为“在每一行前插入 XHTML 换行标记
”。也就是说是“插入”而不是“替换”。但是在使用此函数时，就其效果而言相当于“替换”，因此仍然采用一贯的解释，将其归为字符串替换函数。

通过程序 7-19.php 说明此函数的作用。

程序 7-19.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 7-19.php: nl2br()函数的使用-->
<head>
<title>7-19.php</title>
</head>
<body>
 <form action="7-19.php" method="post">
请输入一段包含回车的文字：<br>
<textarea cols="30" rows="6" name="content"></textarea>
<input type=submit value="提交">
</form>
<?php
  $content=@$_POST["content"];
  //如果用户输入内容不为空
  if($content!=""){
  echo"<hr>";
  echo "直接输出接收到的内容：<br>";
  echo $content;
  echo "<br>(内容长度：".strlen($content).")<br>";
  echo "<hr>";
```

```
    echo "用 nl2br()处理接收到的内容，然后输出：<br>";
    echo nl2br($content);
    echo "<br>(内容长度：".strlen(nl2br($content)).")";
      }
?>
</body>
</html>
```

本程序首先创建了一个 TextArea 多行文本输入框，并要求输入一段包含回车的文字。之所以要求包含回车，就是因为 nl2br 函数处理的对象就是回车。如果不包含回车就无法测试其效果。不妨输入“好好学习，↙天天向上。”，其中“↙”表示按 Enter 键。这时单击“提交看效果”按钮，出现图 7-14 所示的运行结果。



图 7-14　程序 7-19.php 的运行结果

通过图 7-14 可以清楚地看到 nl2br 函数的效果。在未使用 nl2br 函数对接收到的内容进行处理时，本来输入了 3 行内容，在网页中显示时全都连成了一行。这是因为 XHTML 语言不识别回车换行符号，无论在 XHTML 代码中连续输入多少个回车换行，都不会在网页上看到效果，就是因为浏览器会忽略掉 XHTML 代码中的回车换行。用 nl2br()对内容进行处理后，每一行前面都自动添加了一个“
”标记。这个标记就是通常用的 XHTML 中的换行标记“
”，不过是写法略有不同而已。原本输入的三行内容，便正常地显示出来。

虽然 nl2br 函数的本质并没有进行替换，但在使用中，其效果等同于将回车换行符号替换为 XHTML 换行标记。因此在不严格要求的前提下，可以称之为字符替换函数。

2. str_replace 函数

PHP 提供的 str_replace 函数将一个字符串中的任意子串全部替换为另外一个子串，其使用格式如下：

```
mixed str_replace ( mixed search, mixed replace, mixed subject [, int &count] )
```

这个格式看起来有点复杂。简单地解释为，str_replace 函数将 subject 中的所有 search 替换成 replace，并把替换的次数存放在 count 中，其中 count 参数为可选。这里的 search、repalce、subject 以及整个函数的返回值都是 mixed 类型，也就说提供的参数可以是多种类型，常用的有字符串和数组。

程序 7-20.php

```
01  <!--程序 7-20.php：字符串替换函数综合范例-->
02  <?php
03      //单个字符替换
04      $str = "当所有的人[逗]离开我的时候[逗]你劝我要耐心等候[句]";
05      echo "原字符串：<b>".$str."</b><br>";
06      $str = str_replace("[","(",$str);
07      $str = str_replace("]",")",$str);
08      echo "字符替换之后：<b>".$str."</b><br>";
09      //字符串替换
10      $str = str_replace("(逗)","，",$str);
11      $str = str_replace("(句)","。",$str);
12      echo "字符串替换之后：<b>".$str."</b><br>";
13  ?>
```

在程序 7-20.php 中，构造了一个字符串，其中逗号用“[逗]”表示，句号用“[句]”表示。第 10～第 11 行分别进行了两次替换，将字符“[”、“]”分别替换成“(”、“)”。然后输出替换后的字符串。在第 14～第 15 行，又进行了两次替换，将“(逗)”替换成“，”，将“(句)”替换成“。”，然后将最终的字符串输出。本程序运行结果如图 7-15 所示。

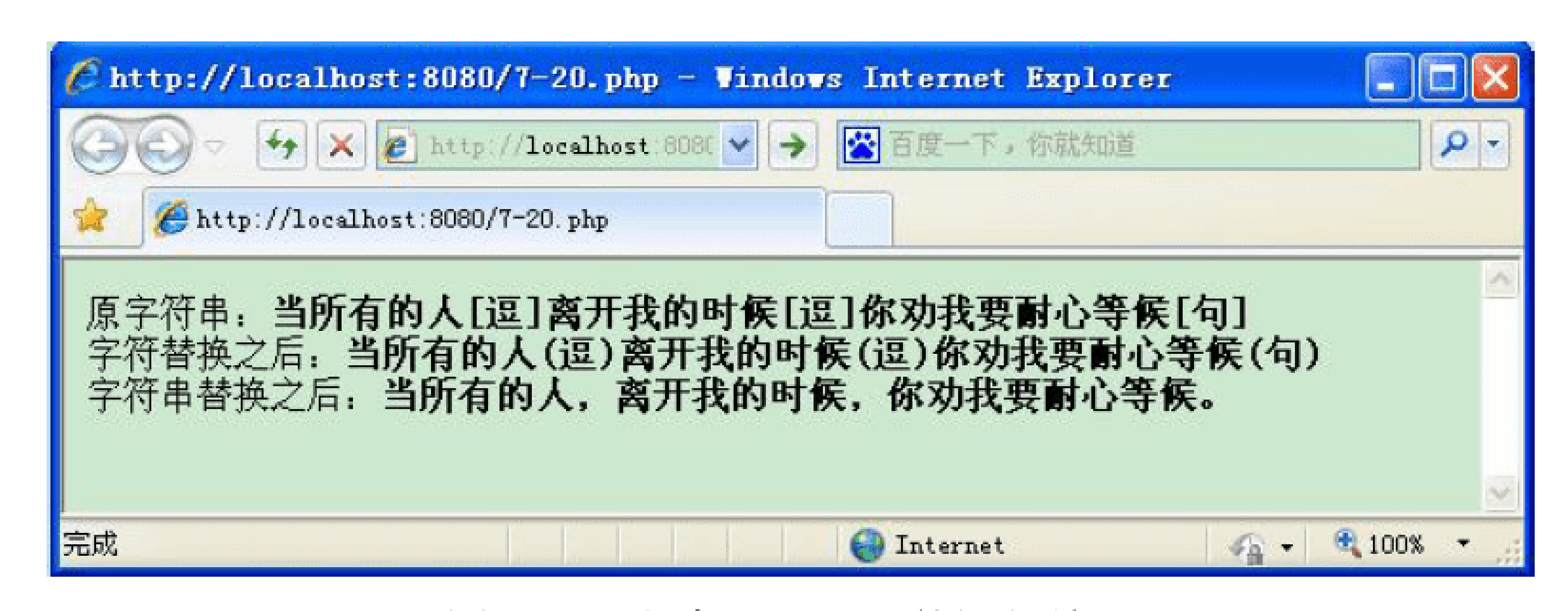


图 7-15 程序 7-20.php 的运行结果

程序 7-20.php 主要用到了 str_replace 函数的普通字符串替换功能。str_replace 函数还可以接收一个数组参数，实现批量的替换。将程序 7-20.php 进行修改，得到程序 7-21.php。

程序 7-21.php

```
<!--程序 7-21.php：字符串替换函数高级应用-->
<?php
     //单个字符替换
    $str = "当所有的人[逗]离开我的时候[逗]你劝我要耐心等候[句]";
    echo "原字符串：<b>".$str."</b><br>";
    $arr1 =  array("[","]");
    $arr2 = array("(",")");
```

```
    $str = str_replace($arr1,$arr2,$str);
     echo "字符替换之后：<b>".$str."</b><br>";
    //字符串替换
    $arr3 = array("(逗)","(句)");
    $arr4 = array("，","。");
    $str = str_replace($arr3,$arr4,$str);
    echo "字符串替换之后：<b>".$str."</b><br>";
?>
```

可以发现，程序 7-21.php 在使用 str_replace 函数时传递了两个数组作为参数，第 1 个数组按顺序存放了要被替换的字符串；第 2 个数组按顺序存放了要替换成的字符串。这样，不论要替换多少个字符串，只要按照顺序分别存放在两个数组中，然后调用 str_repalce 函数即可完成，这样做有明显的优点，在要替换的项目很多的情况下，可以很大程度的简化程序。

本程序的运行结果与 7-20.php 完全相同，运行结果图可参见图 7-15。

7.3.4　字符串截取函数

在编程中经常遇到要将一个字符串的一部分单独取出的情况，也就是字符串的截取。PHP 中常用字符串截取函数有 substr()等。

substr()的使用格式如下：

```
string substr ( string string, int start [, int length] )
```

本函数返回一个字符串中从指定位置开始指定长度的子串。参数 string 为原始字符串，start 为截取的起始位置（从 0 开始计），可选参数 length 为要截取的长度。值得一提的是，参数 start 和 length 均可以用负数，start 为负数时说明从倒数第 start 个字符开始取；length 为负数时表示从 start 位置开始取，向前取 length 个字符结束。

程序 7-22.php

```
<!--程序 7-22.php：字符串的截取-->
<?php
    //构造字符串
    $str = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
     echo "原字符串：<b>".$str."</b><br>";
     //按各种方式进行截取
     $str1 = substr($str,5);
     echo "从第 5 个字符开始取至最后：".$str1."<br>";
     $str2 = substr($str,9,4);
     echo "从第 9 个字符开始取 4 个字符：".$str2."<br>";
     $str3 = substr($str,-5);
    echo "取倒数 5 个字符：".$str3."<br>";
    $str4 = substr($str,-8,4);
    echo "从倒数第 8 个字符开始向后取 4 个字符：".$str4."<br>";
    $str5 = substr($str,-8,-2);
    echo "从倒数第 8 个字符开始取到倒数第 2 个字符为止：".$str5."<br>";
?>
```

本程序运行结果如图 7-16 所示。

可以对照图 7-16 来分析一下整个程序的运行。通过这个例子读者应当对 substr 函数有一个深入的了解。尤其是 start 和 length 两个参数的含义和使用方法，更应该熟练掌握。注意，start 参数为正数时，从 0 开始计数。start 参数为负数时，从 1 开始计数。也就是说没有“倒数第 0 个字符”。读者可以参考本例加深理解，也可以自己动手编制一个程序验证。

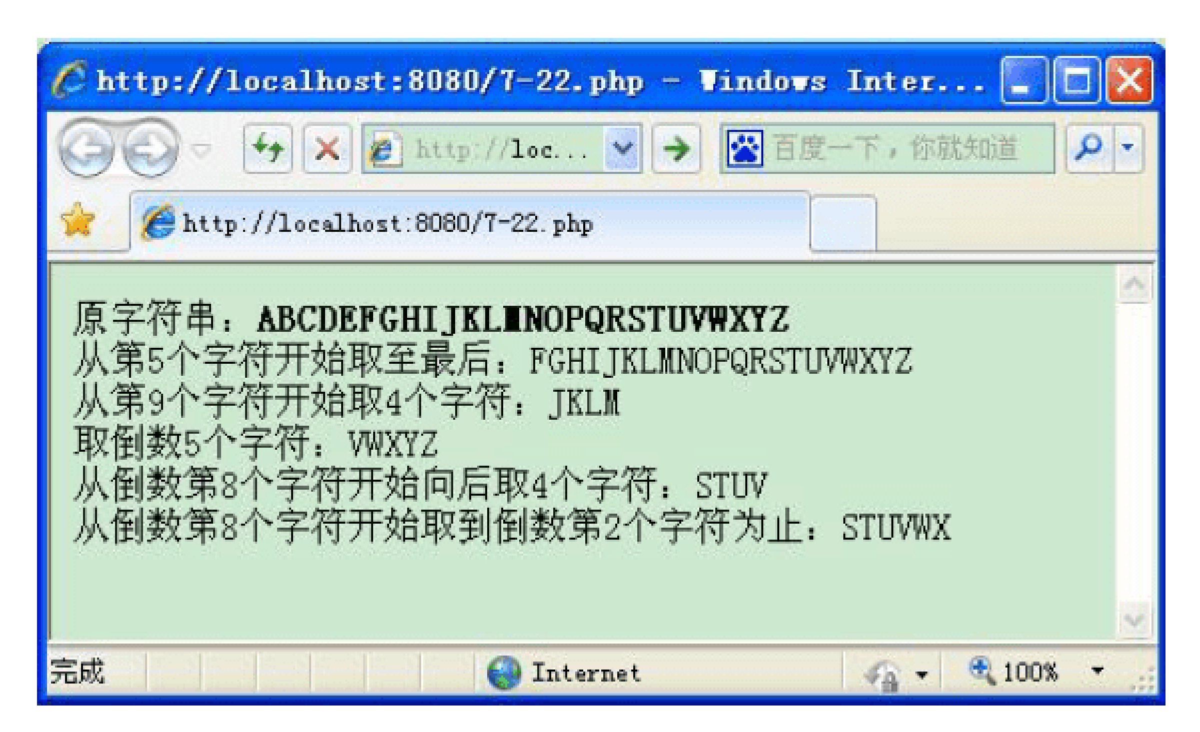


图 7-16　程序 7-22.php 的运行效果

7.3.5　字符串分割函数

在编程中有时候需要将一个字符串按某种规则分割成多个。PHP 提供了 explode()、str_split()等函数完成分割操作。下面分别介绍这两个函数。

1. explode 函数

explode 函数的格式如下：

```
array explode ( string separator, string string [, int limit] )
```

explode 函数用来将一个字符串按照某个指定的字符分割成多段，并将每段按顺序存入一个数组中。该函数的返回值就是一个数组。separator 参数为分割符，可以是一个字符串，也可以是单个字符。string 为要处理的字符串。参数 limit 为可选，如果设置了 limit，则返回的数组包含最多 limit 个元素，最后一个元素将包含 string 的剩余部分。

程序 7-23.php

```
<!--程序 7-23.php：字符串分割-->
<?php
    //构造字符串
     $str = "苹果，空心菜，香蕉，萝卜，大蒜，牛肉";
     echo "原字符串：<b>".$str."</b><br>";
     echo "1.以逗号为分割符分割字符串：<br>";
     $arr1 = explode("，",$str);
     echo "---\$arr1[0]的值：".$arr1[0]."<br>";
     echo "---\$arr1[4]的值：".$arr1[4]."<br>";
     echo "2.分割时指定 limit 参数：<br>";
     $arr2 = explode("，",$str,3);
     echo "---\$arr2[0]的值：".$arr2[0]."<br>";
     echo "---\$arr2[2]的值：".$arr2[2]."<br>";
     echo "---\$arr2[4]的值：".$arr2[4]."<br>";
?>
```

在程序 7-23.php 中，定义了一个普通字符串$str。字符串中出现了多个逗号，用 explode 函数来分隔这个字符串，把“，”作为分割字符。在未提供 limit 参数的情况下，字符串分成 6 小段，并存入数组$arr1 中。每一小段分别对应$arr[0],$arr[1]，…，$arr[5]。然后指定 limit 参数为 3，再次用 explode 函数分隔字符串$str，这时返回的数组$arr2 包含 3 个元素。即$arr2[0],$arr2[1],$arr2[2]。这时$arr2[2]中存放的不是第 3 个逗号之前的内容，而是第 2 个逗号之后的所有内容。程序运行结果如图 7-17 所示。

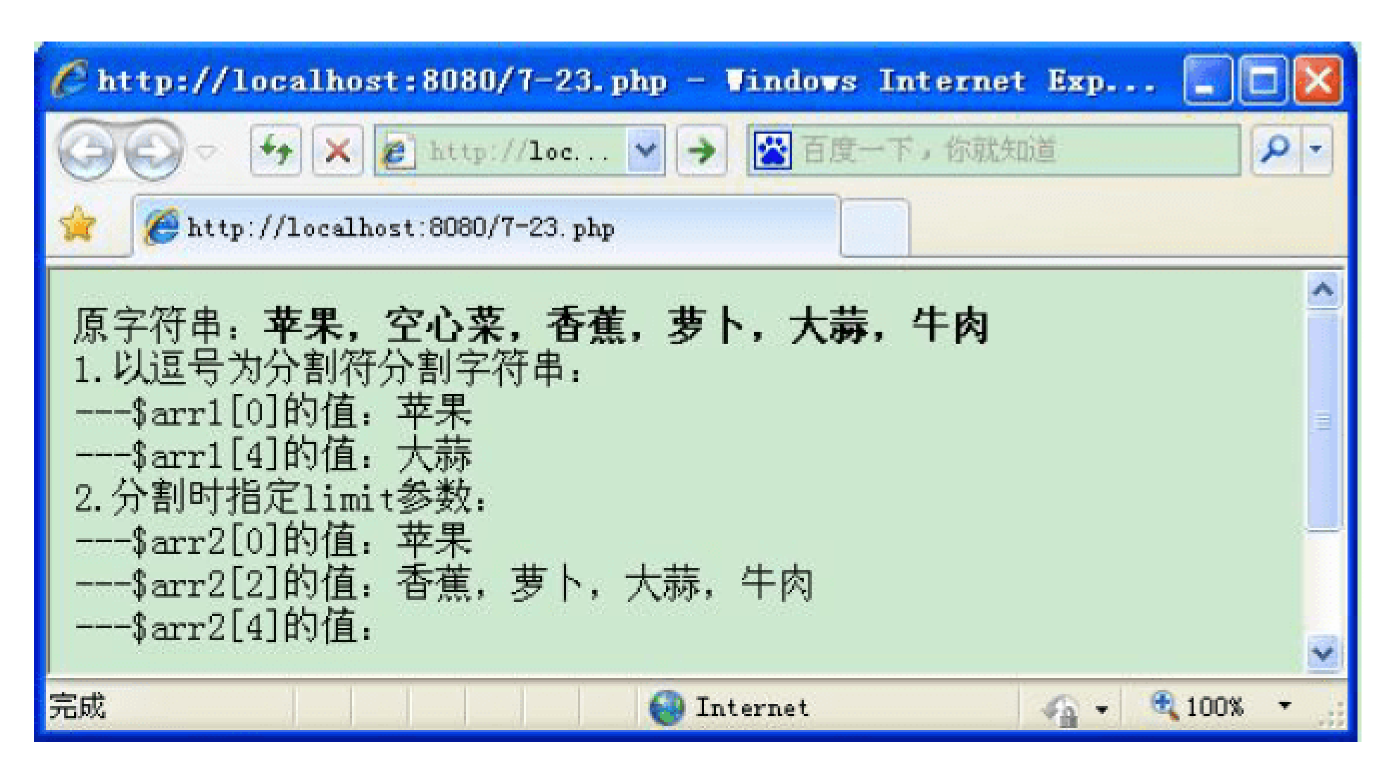


图 7-17　程序 7-23.php 的运行结果

2. str_split 函数

str_split 函数的格式如下：

```
array str_split ( string string [, int split_length] )
```

str_split 函数将一个字符串以一定长度为单位分割成多段，并返回由每一段组成的数组。str_split 函数不是以某个字符串为分割依据，而是以一定长度为分割依据。参数 string 为要分割的字符串，可选参数 length 设置分割的单位长度。

程序 7-24.php

```
<!--程序 7-24.php：字符串分割函数-->
<?php
        //分割英文字符串
        $str = "Quietly I leave,just as quietly I came.";
        echo "原字符串：<b>".$str."</b><br>";
        echo "1.以默认长度分割字符串：<br>";
        $arr1 = str_split($str);
        echo "---\$arr1[0]的值：".$arr1[0]."<br>";
        echo "---\$arr1[1]的值：".$arr1[1]."<br>";
        echo "---\$arr1[10]的值：".$arr1[10]."<br>";
        echo "2.以指定长度 5 分割字符串：<br>";
        $arr2 = str_split($str,5);
        echo "---\$arr2[0]的值：".$arr2[0]."<br>";
        echo "---\$arr2[1]的值：".$arr2[1]."<br>";
        echo "---\$arr2[5]的值：".$arr2[5]."<br>";
        //测试分割中文
        $str2="轻轻地我走了，正如我轻轻地来。";
        echo "原字符串：<b>".$str2."</b><br>";
        echo "1.以指定长度 5 分割字符串：<br>";
```

```
        $arr3 = str_split($str2,5);
        echo "---\$arr3[0]的值: ".$arr3[0]." <br>";
        echo "---\$arr3[1]的值: ".$arr3[1]." <br>";
        echo "2.以指定长度 4 分割字符串: <br>";
        $arr4 = str_split($str2,4);
        echo "---\$arr4[0]的值: ".$arr4[0]." <br>";
        echo "---\$arr4[1]的值: ".$arr4[1]." <br>";
        echo "---\$arr4[4]的值: ".$arr4[4]." <br>";
?>
```

本程序运行结果如图 7-18 所示。

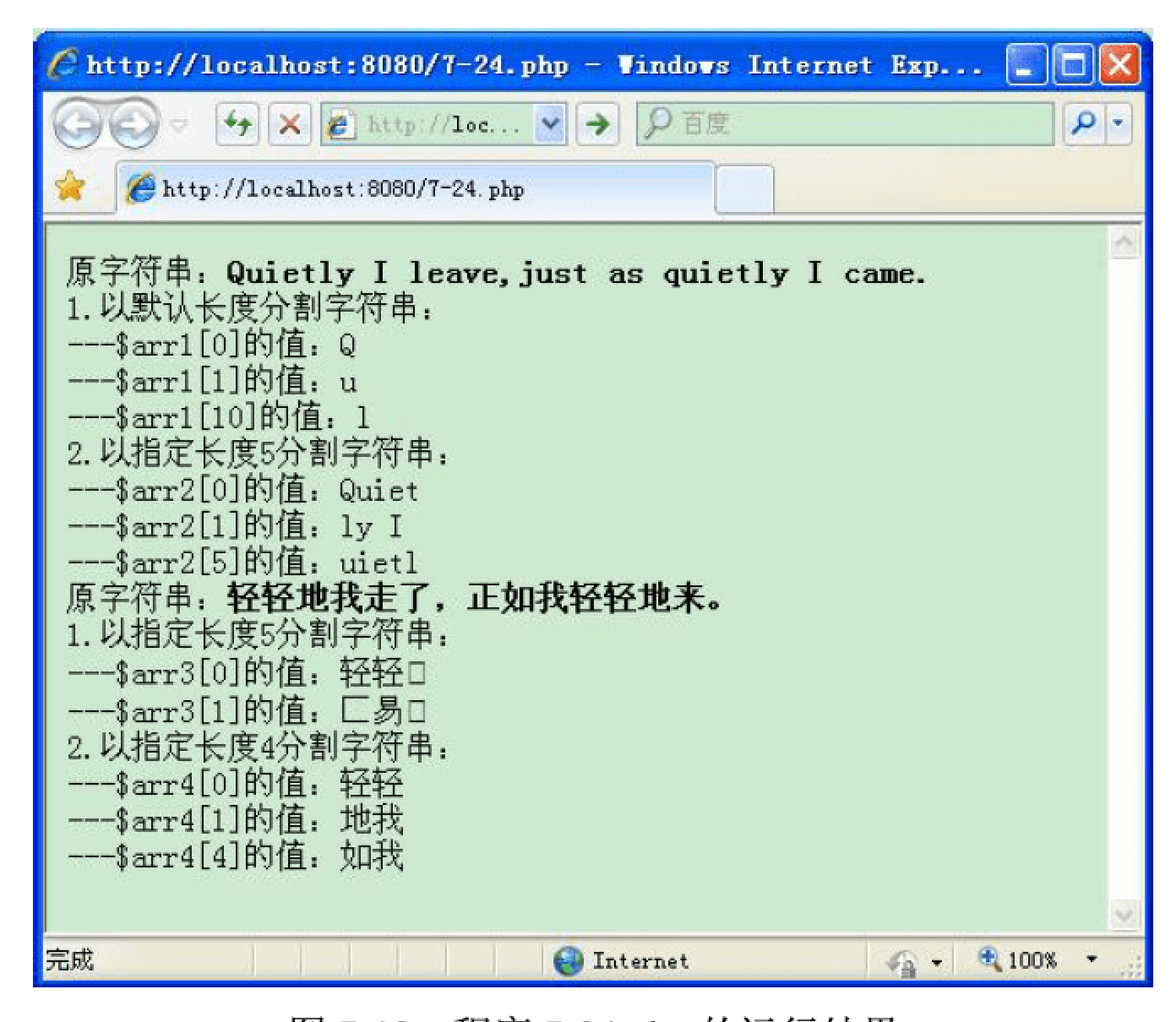


图 7-18　程序 7-24.php 的运行结果

在程序 7-24.php 中，首先构造了一个英文字符串，然后用 str_split 函数直接分割。分割之后字符串被一个字符一个字符的分割开来，并且将这些字符顺次存放到数组$arr1 中。通过图 7-18 的输出结果，能够印证这一点。接下来指定分割的单位为 5，这时候字符串按 5 个字符一段被分割成多段，并存储在数组$arr2 中。这时同样可以看到正确的输出结果。

前面已经提到多次，一个汉字字符占两字节。很多字符串处理函数对中文支持的并不好。为了测试 str_split 函数分割中文的效果，又构造了一个字符串，这个字符串全部由汉字字符构成。首先用 5 作为分割单位来分割字符串，通过输出结果可以看出，str_split 函数无法区别中文，如第 1 段，取出的不是“轻轻地我走”5 个汉字字符，而是“轻轻？”。这里为什么会有一个“？”呢？这个问号不是字符串中的，而是在分割字符串时将第 3 个字“地”分割成了两段，因此无法正确显示这个字符，只能显示为“？”。

为了解决这个问题，又用一个偶数 4 作为分割长度，这时候汉字可以正确显示，整个字符串以 2 个汉字字符为单位被分割成多段。也就是说，在分割中文时，分割长度必须是 2 的倍数，否则将会导致汉字被分成两段而无法正确显示。

使用函数时的分割方案还有不完美之处，当一个字符串是由中英文或中文与阿拉伯数字混合而成，那即使是用 2 的倍数作为分割长度，仍然无法避免汉字被分割的情况。如字

符串“110 是一个重要的电话号码”，如果以 2 或 4 作为分割长度，都会导致“是”这个汉字被分割。因此在使用 str_split 函数时必须充分考虑汉字的影响，否则会产生不可预料的结果。

关于字符串处理函数就介绍到这里。字符串处理函数在编程中使用极为频繁，读者应当熟练掌握，多多积累。本节中介绍的都是字符串处理函数中最为常用的部分，另外还有大量的函数限于篇幅无法一一介绍，读者可以参考表 7-2 及 PHP 官方手册自行学习、掌握，为后面深入学习 PHP 编程打下坚实的基础。

虽然在讲解时每个函数都是独立地讲解，但读者应注意这些函数的结合使用。在一个程序中，可能会同时用到多个函数，通过多个函数的综合应用来实现一个操作，因此读者应在这方面多下功夫。

7.4　图形图像

PHP 提供了一系列函数实现在网站编程中对图形图像进行编辑。虽然使用这些函数能够实现的功能十分有限，无法和功能强大的专业图形图像软件相比，但是在很多需要动态生成图像、自动批量处理图像等方面，能给 PHP 网站开发者带来巨大帮助。其中最为典型的应用有随机图形验证码、图片水印、数据统计中饼状图、柱状图的生成等。表 7-3 中给列出了 PHP 常用的图像处理函数。

表 7-3　PHP 提供的图像处理函数

函　数　名	功　　能
gd_info	取得当前安装的 GD 库的信息
getimagesize	取得图像大小
image_type_to_extension	取得图像类型的文件后缀
imagearc	画椭圆弧
imagechar	水平地画一个字符
imagecharup	垂直地画一个字符
imagecopy	复制图像的一部分
imagecopymerge	复制并合并图像的一部分
imagecopyresized	复制部分图像并调整大小
imagecreatefromgd2	从 GD2 文件或 URL 新建一图像
imagecreatefromgd	从 GD 文件或 URL 新建一图像
imagecreatefromgif	从 GIF 文件或 URL 新建一图像
imagecreatefromjpeg	从 JPEG 文件或 URL 新建一图像
imagecreatefrompng	从 PNG 文件或 URL 新建一图像
imagecreatefromstring	从字符串中的图像流新建一图像
imagecreatetruecolor	新建一个真彩色图像

续表

函 数 名	功 能
imagedashedline	画一虚线
imagedestroy	销毁一图像
imageellipse	画一个椭圆
imagefill	区域填充
imagefilledarc	画一椭圆弧且填充
imagefilledellipse	画一椭圆并填充
imagefilledpolygon	画一多边形并填充
imagefilledrectangle	画一矩形并填充
imagegd2	将 GD2 图像输出到浏览器或文件
imagegd	将 GD 图像输出到浏览器或文件
imagegif	以 GIF 格式将图像输出到浏览器或文件
imageistruecolor	检查图像是否为真彩色图像
imagejpeg	以 JPEG 格式将图像输出到浏览器或文件
imageline	画一条线段
imageloadfont	载入一新字体
imagepalettecopy	将调色板从一幅图像复制到另一幅
imagepng	以 PNG 格式将图像输出到浏览器或文件
imagepolygon	画一个多边形
imagepstext	用 PostScript Type1 字体把字符串画在图像上
imagerectangle	画一个矩形
imagerotate	用给定角度旋转图像
imagesetbrush	设定画线用的画笔图像
imagesetpixel	画一个单一像素
imagesetstyle	设定画线的风格
imagesetthickness	设定画线的宽度
imagesettile	设定用于填充的贴图
imagestring	水平地画一行字符串
imagestringup	垂直地画一行字符串
imagesx	取得图像宽度
imagesy	取得图像高度

PHP 5 提供的图像处理函数总数超过了 100 个，表 7-3 中仅列出了部分常用函数。

PHP 的图像处理函数都封装在一个函数库中，这就是 GD 库。要使用 GD 库中的函数来进行图像处理，必须保证安装了 GD 库。在 PHP 官方的标准发行版本中，都包含了这个

库。如本书介绍的 PHP 5 版本，这个 GD 库存放在 PHP 安装目录下的 ext 子目录下，名为 php_gd2.dll。读者如果担心自己的 PHP 版本是否包含这个函数库，可以打开 PHP 安装目录查找一下。

并不是 php_gd2.dll 库文件存在，就可以使用这些函数了。在默认的 php.ini 设置中，这个库并不自动载入。所以，需要首先打开这个库的自动载入功能，这样这个库中的函数就像 PHP 标准函数一样可以直接在程序中使用了。打开的方法很简单，用记事本打开 php.ini 配置文件，利用查找功能找到“;extension=php_gd2.dll”这一行，将最前面的分号去掉，然后保存，重新启动 IIS（Apache），这时候 GD 库已经被自动加载了，如图 7-19 所示。

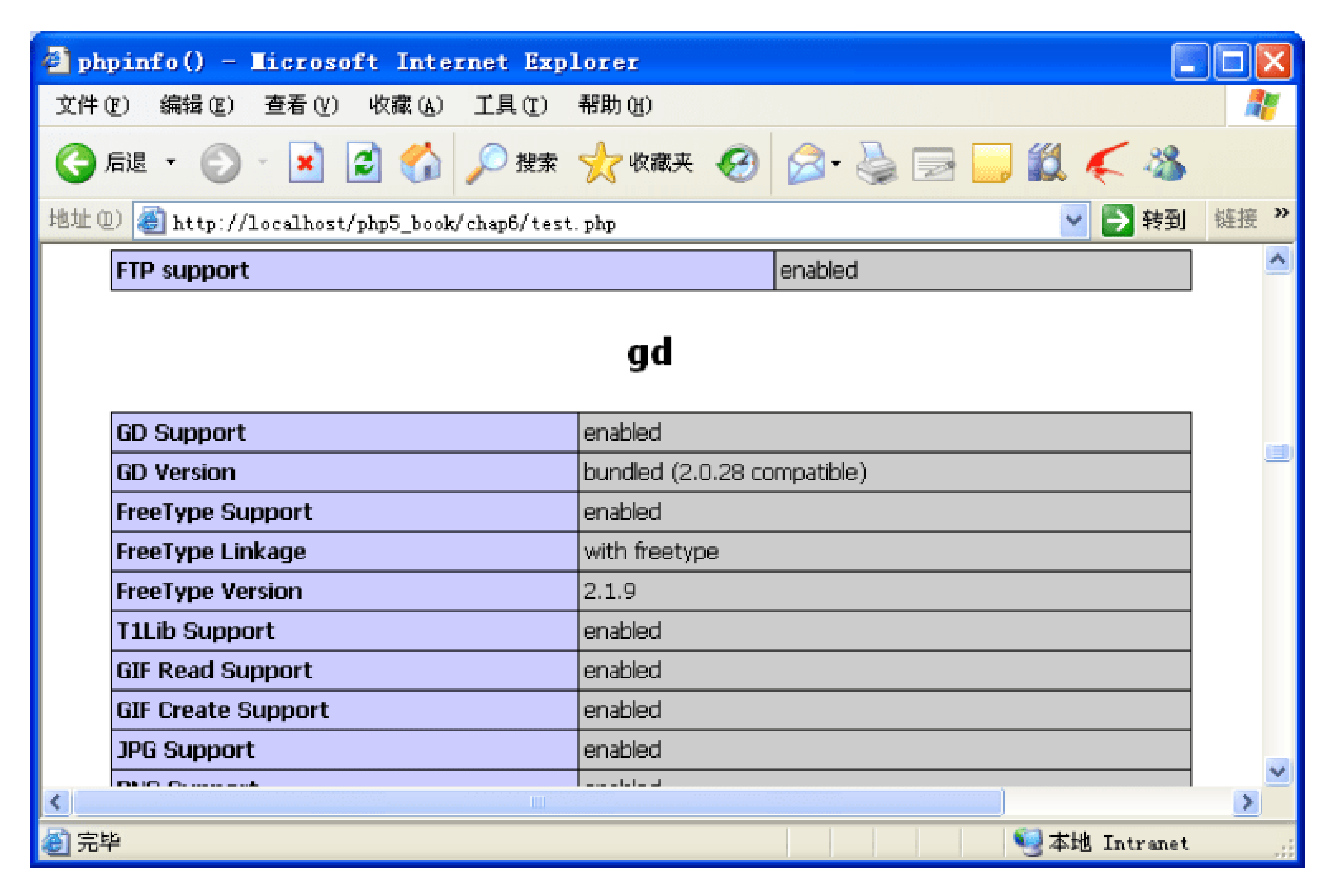
php.ini - 记事本

文件(F)　编辑(E)　格式(O)　查看(V)　帮助(H)

```
;extension=php_curl.dll
;extension=php_dba.dll
;extension=php_dbase.dll
;extension=php_exif.dll
;extension=php_fdf.dll
;extension=php_filepro.dll
extension=php_gd2.dll
;extension=php_gettext.dll
;extension=php_ifx.dll
;extension=php_imap.dll
;extension=php_interbase.dll
;extension=php_ldap.dll
```

图 7-19　开启 GD 图像函数库

若要确定此函数库是否已经被成功加载，可以打开第 2 章中讲过的 PHP 环境信息显示程序，也就是 phpinfo()程序，查看列出的信息中是否有 GD 一项。该项目中详细列出了当前 PHP 的 GD 库信息，如图 7-20 所示。



GD Support	enabled
GD Version	bundled (2.0.28 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.1.9
T1Lib Support	enabled
GIF Read Support	enabled
GIF Create Support	enabled
JPG Support	enabled

图 7-20　GD 库信息

由于本部分函数个数较多，而且具体使用方法较为复杂，要在很有限的篇幅内进行一个较全面的介绍是十分困难的，这里给出两个例子，让读者先体会 PHP 图像处理函数的简单使用。对表中列出的其他函数以及未在表中列出的函数，感兴趣的读者可以参考 PHP 手册进行深入学习。

7.4.1　PHP 基本绘图

通过下面程序 7-25.php 学习有关用 PHP 进行基本绘图的方法。

程序 7-25.php

```
01:  <?php
02:    //程序 7-25.php：图像处理函数使用举例
03:    header("Content-type: image/png");
04:    $im = @imagecreate(200, 100) or die("无法创建图像流");
05:    @imagecolorallocate($im, 240, 150, 255);
06:    $t_color1 = imagecolorallocate($im, 0, 0, 0);
07:    $t_color2 = imagecolorallocate($im,100,100,100);
08:   imagestring($im, 5, 8, 10,  "I like PHP5!", $t_color1);
09:   imagestringup($im,5,8,90,"Hello!",$t_color2);
10:   imageellipse($im,65,65,55,55,$t_color1);
11:   imageellipse($im,65,65,55,55,$t_color1);
12:   imagefilledrectangle($im,110,95,160,30,$t_color2);
13:   imagepng($im);
14:   imagedestroy($im);
15: ?>
```

程序 7-25.php 创建了一幅 PNG 格式的图像，并且在图像上面进行绘图操作。程序运行结果如图 7-21 所示。

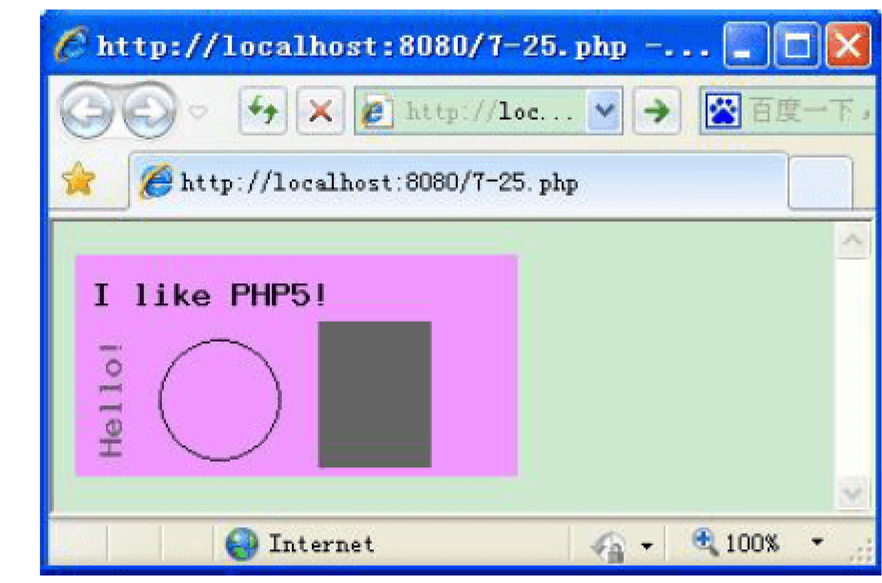


图 7-21　程序 7-25.php 的运行结果

程序 7-25.php 第 3 行指定了图像的类型，即 png 图像，这样程序 7-25.php 虽然是一个 PHP 程序，但是其作用是动态生成一张图像，因此几乎等同于一张图像。在本程序中普通的输出语句如 echo 等都是无效的，这一点读者应当注意。

第 4 行用 imagecreate 函数创建一幅新图像，两个参数为图像的宽度和高度，单位是像素。此函数返回此图像的数据流，存放于$im 变量中。

第 5 行用 imagecolorallocate 函数设置了图像的背景颜色。4 个参数分别为图像流、R 色值、G 色值、B 色值。3 个色值合并即产生了 RGB 色值。这里的 240，150，255 运行之后显示淡紫色。另外（0，0，0）为黑色，（255，255，255）为白色，（255，0，0）为红色等。关于 RGB 颜色的有关详细信息请读者自行查阅有关资料。

第 6～第 7 行分别生成了两种颜色，存在的不同的变量中以备后面使用。第一种为黑色；第二种为浅灰色。

第 8 行用 imagestring 函数在图像上“写入”了一个字符串。6 个参数分别表示图像流、所用字体、写入点的 x 坐标、写入点的 y 坐标、要写入的字符串、字符串颜色。注意：一

是函数的第二个参数取值范围为 1～5，分别代表了不同大小和是否加粗的 5 种字体，读者可以试着修改此参数来观察程序运行效果；二是这里的 x,y 坐标都是相对于图像的左上角，最左上角坐标为（0,0），向右为 x 轴，向下为 y 轴。单位都是像素。

第 9 行用 imagestringup 函数向图像中竖向写入一个字符串。函数的参数含义与 imagestring 函数相同。

第 10 行用 imageellipse 函数在图像中绘制了一个圆。函数第一个参数为图像流，最后一个为绘图所用颜色，第 2～第 5 个参数分表表示圆心的 x 坐标、圆心的 y 坐标、圆的 x 方向半径长度、圆的 y 方向半径长度。在本例中，绘制了一个圆心在（65,65）这个点，半径为 55 的正圆。如果要绘制一个椭圆，只需要确定圆心位置，然后分别设置 x 方向半径和 y 轴方向半径即可。当这两个半径相等时是一个圆，不相等时是一个椭圆。

第 11 行用 imagefilledrectangle 函数绘制了一个矩形，并对矩形进行颜色填充。第一个参数为图像流，最后一个参数为填充颜色。第 2～第 5 个参数的含义分别为矩形左上角 x 坐标、矩形左上角 y 坐标、矩形右下角 x 坐标、矩形右下角 y 坐标。也就是说只要提供矩形的左上角和右下角坐标，即可绘制此矩形。

第 12 行用 imagepng 函数将此图像流输出为一张 png 格式的图片。也就是在浏览器中看到的图片。

第 13 行销毁了这个图像流。

在本例中除了 png 格式，还可以把图像输出为 jpg、gif 等常用的格式，只需要更改一下程序中第 3 行所指定的图像类型即可。

7.4.2　网站图形验证码制作

图像验证码程序是当前 Web 开发中常用的程序。本章学习了 PHP 的图像处理函数，结合前面章节学习的 Session 函数以及表单数据提交技术，可以写出一个完整的图像验证码程序。

验证码在网站中的作用一般是防止恶意“灌水”，也就是恶意发布垃圾信息。如果没有验证码，攻击者可以利用辅助软件实现自动提交、自动注册等。由于软件执行的效率高、速度快且可以连续工作，因此常用来被攻击某个网站，制造大量垃圾数据，严重影响网站正常运行。

采用了验证码的方式之后，由于验证码每次都不一样，只有验证码输入正确才能提交信息，这样辅助软件就无法随意向服务器提交信息了。因此，验证码的设计也有一些原则，如验证码的生成是随机的，无规律可循。另外，有的辅助软件有文字识别功能，能够从图片中辨析出文字，因此验证码中的数字可以采用随机的颜色，而且七扭八歪不易辨认。总之，最理想的验证码应该是人的肉眼可以很容易地辨认出来，但是用软件识别极为困难。

鉴于此，在设计这个验证码程序时，就不是简单的创建一幅图片，然后随机生成几个数字，再加入一些干扰。用 PHP 提供的图像处理函数，可以在图像上加入一些密密麻麻的像素点，然后随机绘制两条虚线，再将几个数字的位置打乱。这样，计算机识别就变得十分困难了。

本实例用到以下 3 个文件。

程序 7-26-showing.php：生成验证码，将验证码写入图片，并输出图片。

程序 7-26-login.html：调用 showimg.php，将用户输入的验证码提交到 check.php 进行验证。

程序 7-26-check.php：用来验证用户输入的验证码是否正确。

下面就来看一下具体的代码。

程序 7-26-showing.php

```
<?php
<!--程序 7-26-showing.php：生成验证码图片，并输出-->
session_start();                                    //启动 session
header('Content-type: image/gif');                  //输出头信息
$image_w=90;                                        //验证码图形的宽
$image_h=25;                                        //验证码图形的高
$num=range(0,9);
$string="";                                         //初始化
$len=count($num);                                   //新数组的长
for($i=0;$i<4;$i++)
{
    $new_num[$i]=$num[rand(0,$len-1)];              //在$num 数组中随机取出 4 个字符
    $string=$string.$new_num[$i];                   //生成验证码字符串
}
$_SESSION['string']=$string;                        //使用$_SESSION 变量传值
$image=imagecreatetruecolor($image_w,$image_h);         //创建图片对象
$white=imagecolorallocate($image, 255, 255, 255);
$pt_color=imagecolorallocate($image, mt_rand(0,255),mt_rand(0,255),
mt_rand(0,255));
imagefill($image,0,0,$white);                       //设置背景颜色为白色
for($i=0;$i<100;$i++)                               //加入 100 个干扰的彩色点
{
imagesetpixel($image, rand(0,$image_w), rand(0,$image_h),$pt_color);
}
for($i=0;$i<count($new_num);$i++)                   //在背景图片中循环输出 4 位验证码
{
    $x=mt_rand(1,8)+$image_w*$i/5;                  //设定字符所在位置 X 坐标
    $y=mt_rand(1,$image_h/5);                       //设定字符所在位置 Y 坐标
    //随机设定字符颜色
$color=imagecolorallocate(
         $image,mt_rand(0,255),mt_rand(0,255),mt_rand(0,255));
    //输入字符到图片中
    imagestring($image,5,$x,$y,$new_num[$i],$color);
}
    imagepng($image);
    imagedestroy($image);
?>
```

程序中的重要位置都已经做了注释，再此不再详细讲解。本程序运行后可以在浏览器中生成一张带有验证码的图片。每次刷新程序都会生成一个新验证码。

程序 7-26-login.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 7-26-login.html：图形验证码程序-->
<head>
 <title>7-26-login.html</title>
</head>
<body>
 <form action="7-26-check.php" method="post">
 <img src="7-26-showing.php"><br>
 请输入验证码：<input type="text" name="passcode">
 <input type=submit value="确定">
 </form>
</body>
</html>
```

本程序是一段纯 HTML 代码，无须多做解释。值得注意的是，在调用这个图片时，采用“<img src= “7-26-showimg.php” >”的方式。因为验证码图片本身是一张图片，所以使用<img>标签引用。这张图片又是用 PHP 程序生成的，因此直接用“src= 7-26-showimg.php”调用。运行结果如图 7-22 所示。



图 7-22 程序 7-26-login.html 的运行结果

```
01 <?php
02    //验证用户输入的验证码是否正确
03        session_start();
04        $passcode=$_SESSION["Checknum"];
05        $usercode=$_POST["passcode"];
06      if($passcode == $usercode){
07        echo "验证码正确！验证通过！";
08     }else{
09        echo "验证码输入错误！验证失败！";
10    }
11 ?>
```

程序第 3 行是将 Session 中存储的正确的验证码读取出来。第 4 行将用户输入的验证码接收过来。然后进行比较，如果相等，则说明用户输入的验证码正确；否则不正确。

7.4.3 图片水印制作

不仅可以直接创建一个图像流绘制图形，还可以将一张已有的图片作为图像流读入，然后在此基础上对图像进行处理。这一功能常用来制作图像水印。所谓图像水印，就是在

图像上标上一些特殊的图形或符号，用来作为图像所有者的标志或防止图片被盗用。下面就看一个这样的例子。

本例使用了一张原始图片 pic.jpg，现在用 PHP 将此图片进行处理，在图片表面按一定规律加上文字标签，产生水印效果，使之不能被直接盗用。

程序 7-27.php

```
01 <?php
02     //程序 7-27.php：为图片加水印
03     header("Content-type: image/jpeg");
04     $im = imagecreatefromjpeg("pic.jpg");
05     $white = imagecolorallocate($im,255,255,255);
06     $width=imagesx($im);
07     $height=imagesy($im);
08     $x=0;
09     $y=0;
10     while($x<$width && $y<$height){
11      imagestring($im,2, $x,$y,"http://www.xxx.com", $white);
12      $x+=20;
13     $y+=20;
14     }
15     imagejpeg($im);
16     imagedestroy($im);
17 ?>
```

本程序第 3 行设定本页面输出类型为 jpeg 图像。

第 4 行用 imagecreateformjpeg 函数打开了一张图片 pic.jpg，并返回此图片的数据流。

第 5 行定义了一个颜色（白色）。

第 6～第 7 行用 imagesx 和 imagesy 函数取得图片 pic.jpg 的原始尺寸。

第 8～第 9 行定义了用于控制文字添加位置的两个变量。

第 10～第 14 行用循环向图片中添加多行文字，用$x 和$y 两个变量控制位置和循环次数。

第 15 行输出此图片，第 16 行销毁数据流。

程序运行前和运行后的图像分别如图 7-23 和图 7-24 所示。

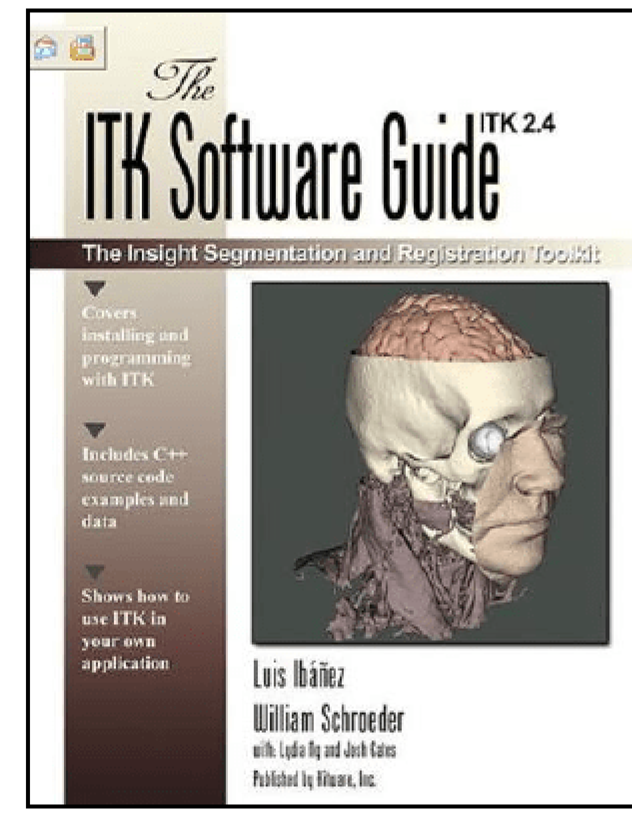


图 7-23　原始图片

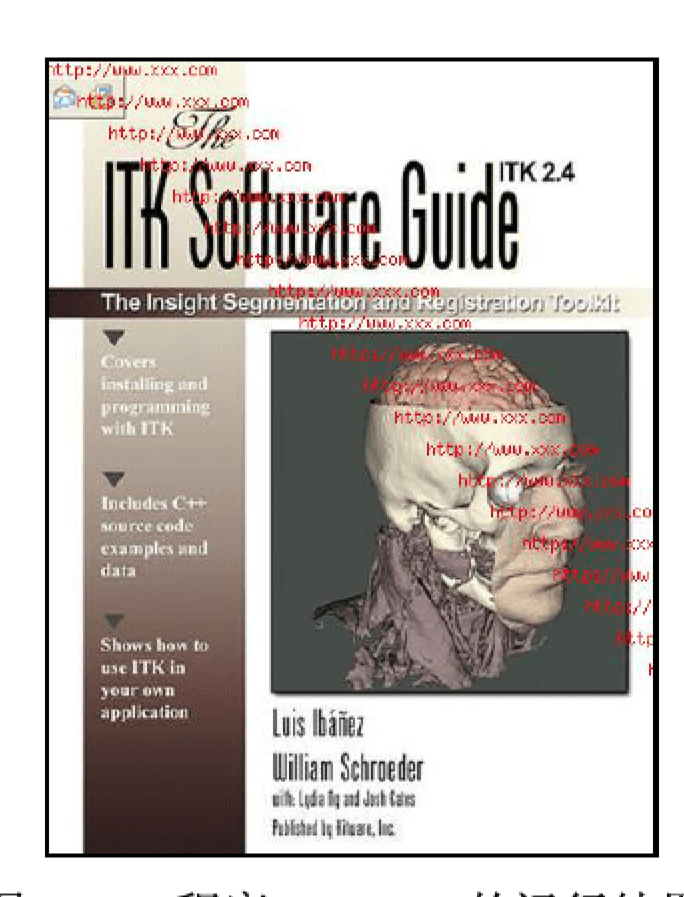


图 7-24　程序 7-27.php 的运行结果

可以看到，处理后的图片上加入了文字标记，这就基本达到了处理意图。但是同时也可以看出，处理后的图片由于文字的加入影响了图像的观赏性。因此水印如何加，加在什么位置，即能起到水印的作用，又不严重影响美观，才是在处理中最应考虑的问题。

PHP 的图像处理函数就介绍这些，希望读者对此有一个基本的了解，为以后深入学习打下基础。

7.5　时间与日期

时间日期函数用来获取服务器的时间和日期，或对时间日期类型的数据进行各种处理，来满足程序的需要。在编程中时常要用到时间日期，如信息发布时记录发布的时间、用户注册时要记录注册的时间、记录用户进行某些操作的时间等。PHP 5 提供的时间日期函数如表 7-4 所示。

表 7-4　PHP 提供的时间与日期处理函数

函　数　名	功　　能
checkdate	验证一个格里高里日期
date_default_timezone_get	取得一个脚本中所有日期时间函数所使用的默认时区
date_default_timezone_set	设定用于一个脚本中所有日期时间函数的默认时区
date_sunrise	返回给定的日期与地点的日出时间
date_sunset	返回给定的日期与地点的日落时间
date	格式化一个本地时间/日期
getdate	取得日期/时间信息
gettimeofday	取得当前时间
gmdate	格式化一个 GMT/UTC 日期/时间
gmmktime	取得 GMT 日期的 UNIX 时间戳
gmstrftime	根据区域设置格式化 GMT/UTC 时间/日期
idate	将本地时间日期格式化为整数
localtime	取得本地时间
microtime	返回当前 UNIX 时间戳和微秒数
mktime	取得一个日期的 UNIX 时间戳
strftime	根据区域设置格式化本地时间/日期
strptime	解析由 strftime()生成的日期/时间
strtotime	将任何英文文本的日期时间描述解析为 UNIX 时间戳
time	返回当前的 UNIX 时间戳

通过表 7-4 可以看到，PHP 提供了很多函数实现各种时间/日期操作。其中，不乏很有趣的函数，如返回某给定日期与地点的日出/日落时间。不过其中部分函数并没有很大的实用价值，只需要熟练掌握其中几个函数的使用，即可实现绝大多数常见的应用。

7.5.1 获取当前时间的 UNIX 时间戳

很多读者可能不明白什么是 UNIX 时间戳。UNIX 时间戳是指从 UNIX 纪元（格林威治时间 1970 年 1 月 1 日 00 时 00 分 00 秒）开始到当前时间为止相隔的秒数。因此，很显然 UNIX 时间戳应该代表一个很大的整数。UNIX 时间戳在很多时候非常有用，尤其在对时间进行加减时作用最为明显。如当前时间是“2006 年 10 月 10 日 10 点 10 分 10 秒”，在这个时间基础上加上 25 天 8 小时 55 分 58 秒，会得到一个什么时间呢？可能推算起来比较复杂。因为除了时间进位以外，还涉及不同月份天数可能不同（可能是 28 天、29 天、30 天、31 天）。所以用数学方法直接加减是不行的。如果使用 UNIX 时间戳，在第一个时间的基础上加上一定的秒数，得到的就是第二个时间的 UNIX 时间戳。然后用 PHP 的有关函数把这个时间戳转换成普通时间格式显示即可。

PHP 中提供了 time 函数来直接获取当前时间的 UNIX 时间戳。

程序 7-28.php

```
<!--程序 7-28.php：获取 UNIX 时间戳-->
<?php
$tm= time();
echo "当前时间的 UNIX 时间戳为：".$tm;
?>
```

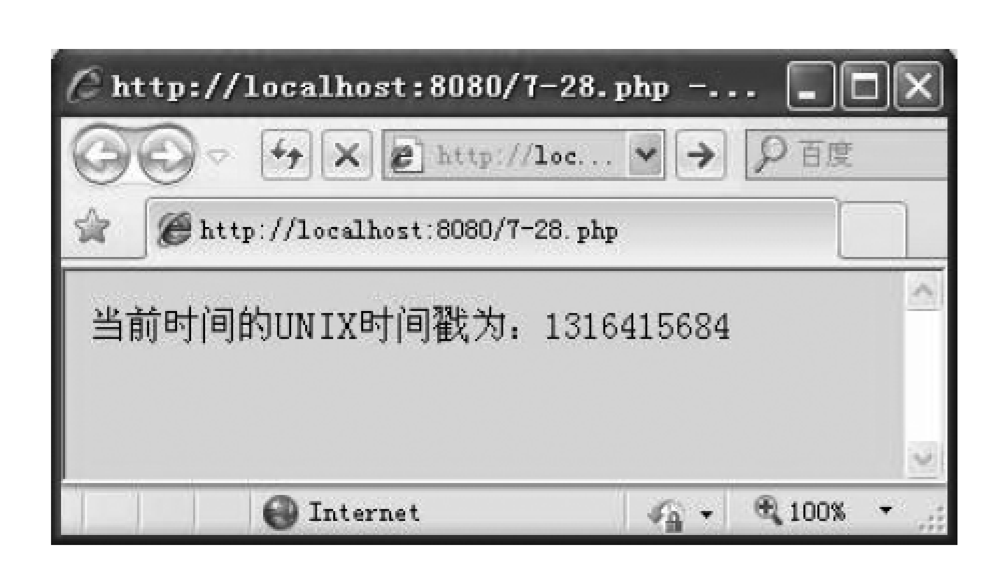


图 7-25 程序 7-28.php 的运行结果

本程序运行结果如图 7-25 所示。

数字 1316415684 表示的是从 1970 年 1 月 1 日 0 点 0 分 0 秒到本程序执行时相隔的秒数。如果每隔一段时间刷新一下页面，会发现时间戳的值每次都会变化。因为每过一秒钟，这个数字就增加 1。每次刷新页面都会重新调用程序，都会获得一个不同的时间，因此 UNIX 时间戳在不断变化。具体地说，这个数字在不断增大。

7.5.2 获取指定时间的 UNIX 时间戳

能不能获得一个指定时间的 UNIX 时间戳呢？也就是说要获得的时间戳不是当前时间的，而是一个固定时间的。PHP 给提供了 mktime 函数和 strtotime 函数完成这个操作。

mktime 函数的格式如下：

```
int mktime ( [int hour [, int minute [, int second [, int month [, int day
[, int year]]]]]] )
```

本函数的作用是根据给出的参数返回 UNIX 时间戳。6 个参数全都是整数，分别代表小时、分钟、秒、月、日、年。参数可以从右向左省略，任何省略的参数会被设置成本地日期和时间的当前值。当全部参数都被省略时，获得的就是当前时间的 UNIX 时间戳。

另外 strtotime 函数允许使用一个时间字符串作为参数来获取 UNIX 时间戳。这个时间串的顺序与中文习惯较为吻合。如“2000-11-12 10:34:55”表示 2000 年 11 月 12 日 10 时

34 分 55 秒。该字符串指代了一个具体的时间，可以作为 strtotime 函数的参数，来获得这个时间的 UNIX 时间戳。

程序 7-29.php

```
<!--程序 7-29.php：获取指定时间的 UNIX 时间戳-->
<?php
//用 mktime()返回时间戳
$tm= mktime(23,56,59,07,20,2011);
echo "2011 年 07 月 20 日 23 点 56 分 59 秒的 UNIX 时间戳为：".$tm;
//用 strtotime()返回时间戳
$tm2= strtotime("2011-07-20 23:56:59");
echo "<br>用 strtotime 获得的同一时间的时间戳：".$tm2;
?>
```

本程序运行结果如图 7-26 所示。

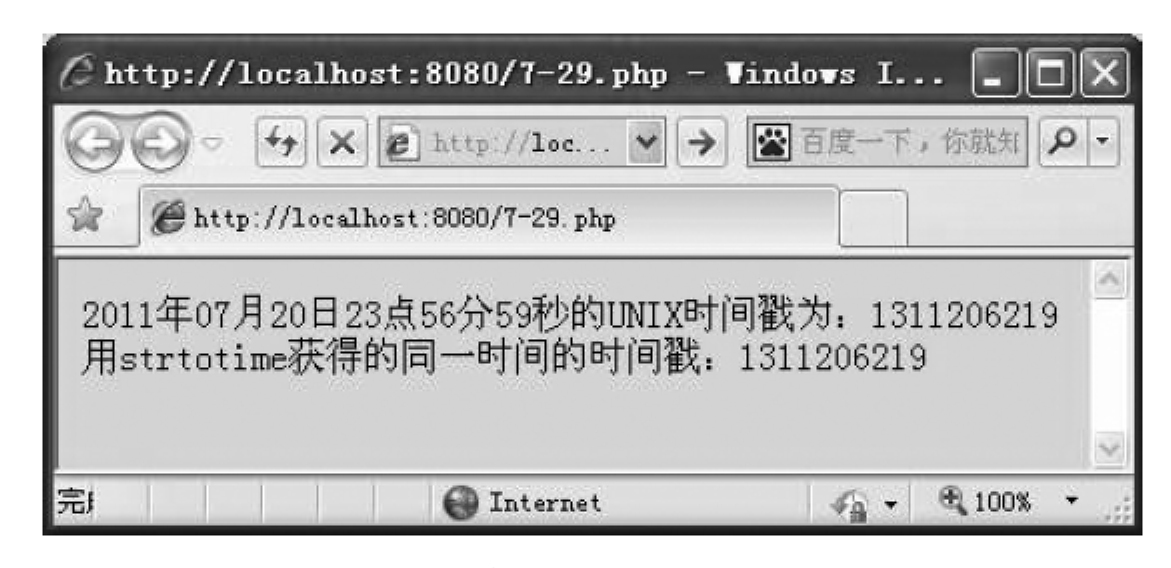


图 7-26　程序 7-29.php 的运行结果

程序 7-29.php 用 mktime 函数将一个指定时间格式化为 UNIX 时间戳。然后，用 strtotime 函数同样格式化这个时间，不过用一个字符串而不是 6 个数字作为参数。运行结果显示两个函数的返回结果完全相同。刷新页面时，时间戳值都不会发生变化。因为，这个时间戳是一个固定的时间，不会随当前时间变化而变化。

有的读者可能会问一个问题：如果使用 mktime 函数时提供的参数不符合常规，会出现什么情况呢？例如，每年最多有 12 个月，不可能有 14 月，如果给月份这个参数提供一个 14，会怎样呢？实际上，PHP 会把某年的 14 月作为下一年的 2 月，如 1999 年 14 月会被认为是 2000 年 2 月。同理，如果时间设置为 23 分 70 秒，那 PHP 会作为 24 分 10 秒来处理。读者可以自行编制程序进行测试。

7.5.3　取得时间日期信息

前面学习了如何获得一个时间的 UNIX 时间戳。虽然用 UNIX 时间戳有利于在计算机中进行时间的计算，但是在显示时间时还是应该显示成通用的“年月日时分秒”以及星期几等格式，而不是直接输出一个 UNIX 时间戳。PHP 中提供了 date()和 getdate()等函数实现从 UNIX 时间戳到通用时间日期的转换。

1. getdate 函数

getdate 函数用来将一个 UNIX 时间戳格式化成具体的时间日期信息，其使用格式如下：

```
array getdate ( [int timestamp] )
```

其中，参数 timestamp 就是一个 UNIX 时间戳。如果不指定参数，则默认使用当前时间。该函数返回一个数组，数组中存放了详细的时间信息。通过数组下标可以取得数组中的元素值。其下标与值的对应关系如表 7-5 所示。

表 7-5　数组下标与值的对应关系

下　　标	说　　明	返回值例子
seconds	秒的数字表示	0～59
minutes	分钟的数字表示	0～59
hours	小时的数字表示	0～23
mday	月份中第几天的数字表示	1～31
wday	星期中第几天的数字表示	0（表示星期天）～6（表示星期六）
mon	月份的数字表示	1～12
year	4 位数字表示的完整年份	如：1999 或 2003
yday	一年中第几天的数字表示	0～365
weekday	星期几的完整文本表示	Sunday～Saturday
month	月份的完整文本表示	January～December
0	该时间的 UNIX 时间戳	整数

下面看程序 7-30.php，全面展示该函数的强大功能。

程序 7-30.php

```
01  <!--程序 7-30.php：getdate 函数获取详细的时间信息-->
02  <?php
03    //首先假设一个时间
04    $dt= "2011-07-01 08:30:00";
05    echo "时间：".$dt."<br>";
06    //将此时间格式化为 UNIX 时间戳
07    $tm= strtotime($dt);
08    echo "此时间的 UNIX 时间戳：".$tm."<br>";
09    //获取该时间的详细信息
10    $arr = getdate($tm);
11    //输出详细信息
12    echo "秒：".$arr["seconds"]."<br>";
13    echo "分：".$arr["minutes"]."<br>";
14    echo "时：".$arr["hours"]."<br>";
15    echo "日：".$arr["mday"]."<br>";
16    echo "月：".$arr["mon"]."/".$arr["month"]."<br>";
17    echo "年：".$arr["year"]."<br>";
18    echo "星期：".$arr["wday"]."/".$arr["weekday"]."<br>";
19    echo "该日期是该年中的第".$arr["yday"]."天<br>";
20  ?>
```

本程序中，第 4 行设置了一个时间，第 7 行将此时间格式化成 UNIX 时间戳。第 10 行将此时间戳用 getdate 函数获取详细时间信息。然后在第 12～第 19 行分别输出了全部的时间信息。程序的输出结果如图 7-27 所示。

本程序中假定日期为“2011-07-01 08:30:00”，实际上可以直接用语句“$arr = getdate();”获得当前时间的详细信息。这时输出的时间信息就是当前程序执行时的时间信息。感兴趣的读者可以自行测试。



图 7-27　程序 7-30.php 的运行结果

2. date 函数

date 函数用来将一个 UNIX 时间戳格式化成指定的时间/日期格式。getdate 函数可以获取详细的时间信息，但是很多时候并不是要取得如此具体的时间信息，而是将一个 UNIX 时间戳所代表的时间按照某种容易识读的格式输出。这就需要用到 date 函数。该函数的使用格式如下：

```
string date ( string format [, int timestamp] )
```

该函数直接返回一个字符串。这个字符串就是一个指定格式的日期时间。参数 format 是一个字符串，用来指定输出的时间的格式。可选参数 timestamp 是要处理的时间的 UNIX 时间戳。如果参数为空，那么默认值为当前时间的 UNIX 时间戳。

本函数的重点是学习如何使用 format 参数。format 参数必须由指定的字符构成，不同的字符代表不同的特殊含义，如表 7-6 所示。

表 7-6　format 参数一览表

format 参数	说　　明	返回值例子
d	月份中的第几天，有前导零的 2 位数字	01～31
D	星期中的第几天，文本表示，3 个字母	Mon～Sun
j	月份中的第几天，没有前导零	1～31
l（小写“L”）	星期几，完整的文本格式	Sunday～Saturday
N	ISO-8601 格式数字表示的星期中的第几天（PHP 5.1.0 中的新参数）	1（表示星期一）～7（表示星期天）
w	星期中的第几天，数字表示	0（表示星期天）～6（表示星期六）
z	年份中的第几天	0～366
W	ISO-8601 格式年份中的第几周，每周从星期一开始（PHP 4.1.0 新加的）	如 42（当年的第 42 周）

续表

format 参数	说　　明	返回值例子
F	月份，完整的文本格式	January～December
m	数字表示的月份，有前导零	01～12
M	3 个字母缩写表示的月份	Jan～Dec
n	数字表示的月份，没有前导零	1～12
t	给定月份所应有的天数	28～31
L	是否为闰年	如果是闰年为 1，否则为 0
Y	4 位数字完整表示的年份	如 1999 或 2003
y	2 位数字表示的年份	如 99 或 03
a	小写的上午和下午值	am 或 pm
A	大写的上午和下午值	AM 或 PM
B	Swatch Internet 标准时	000～999
g	小时，12 小时格式，没有前导零	1～12
G	小时，24 小时格式，没有前导零	0～23
h	小时，12 小时格式，有前导零	01～12
H	小时，24 小时格式，有前导零	00～23
i	有前导零的分钟数	00～59
s	秒数，有前导零	00～59
e	时区标识（PHP 5.1.0 中的新参数）	如 UTC、GMT、Atlantic/Azores
I	是否为夏令时	如果是夏令时为 1，否则为 0
O	与格林威治时间相差的小时数	如+0200
T	本机所在的时区	如 EST，MDT 等

表 7-6 中列出了绝大部分 format 参数字符，极为不常用的没有列出。通过表 7-6 已经看出 format 字符数量众多，涉及方方面面。date 函数可以完成的功能极强。

下面通过一个例子来看这些 format 字符如何使用。

程序 7-31.php

```
<!--程序 7-31.php：用格式化字符串输出时间信息-->
<?php
  //设置一个时间（如采用当前时间可用 time()）
  $tm = strtotime("2011-09-09 10:30:40");
  echo "初始化设置的时间为：2011-09-09 10:30:40<br>";
  //使用不同的格式化字符串测试输出效果
  echo date("Y-M-D H:I:S A",$tm)."<br>";
  echo date("y-m-d h:i:s a",$tm)."<br>";
  echo date("Y年m月d日[l] H点i分s秒",$tm)."<br>";
  echo date("F,d,Y l",$tm)."<br>";
  echo date("Y-M-D H:I:S",$tm)."<br>";
  echo date("Y-M-D H:I:S",$tm)."<br>";
```

```
    echo date("所在时区：T，与格林威治时间相差：O 小时",$tm)."<br>";
    //输出详细信息
   ?>
```

本程序的运行结果如图 7-28 所示。

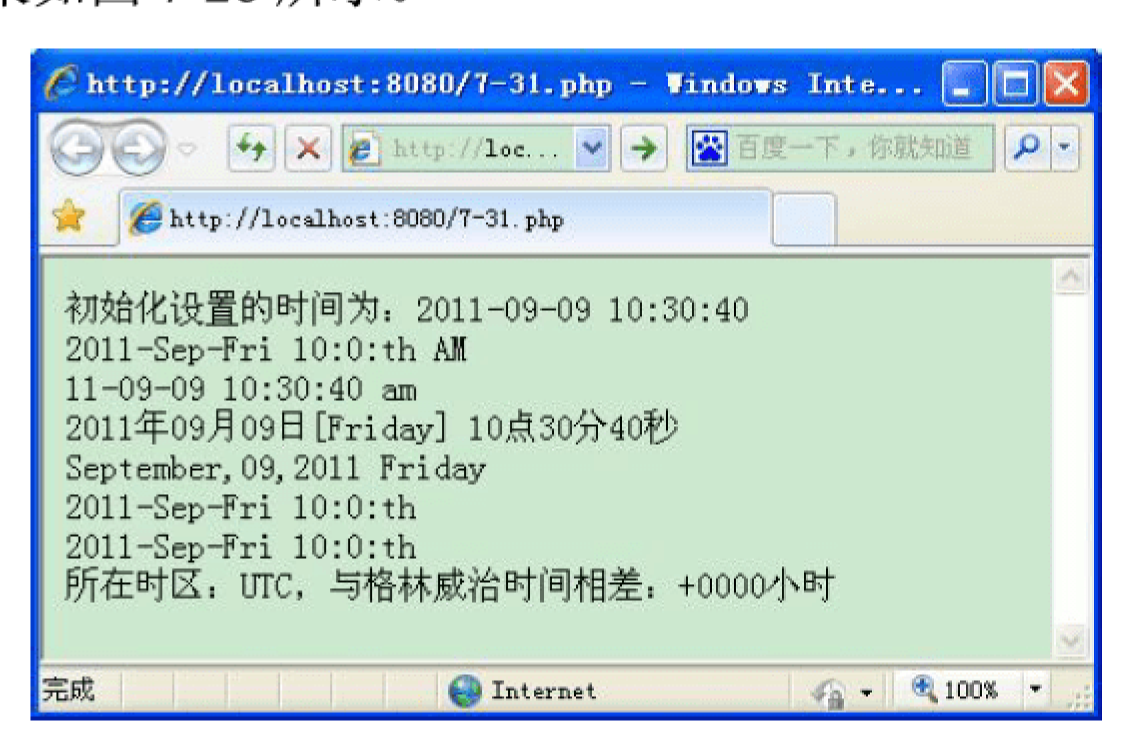


图 7-28　程序 7-31.php 的运行结果

通过程序 7-31.php 可以看出，格式化字符串的使用非常灵活。只要在字符串中包含相关字符，date 函数就能把这些字符替换成指定的时间日期信息。可以利用这个函数随意输出需要的时间日期格式。

程序的最后一条输出用的是格式字符 T 和 O 输出运行本程序服务器所处的时区以及本时区和格林威治标准时间相差的小时数。程序输出时区为 UTC，相差时间为 0 小时。这虽然与世界标准时区和时间相符，但是并不是本地的时间。例如，北京时间要比格林威治时间晚 8 个小时，因此在取得的本地时间基础上再增加 8 个小时才是北京时间。增加 8 个小时的方法很简单，在已经取得的当前时间的 UNIX 时间戳上加 8×60×60 即是 8 小时之后的时间戳。如果读者在编写程序时发现程序获得的时间与北京时间不符，应该考虑是否是时区问题，对取得的时间进行相应处理即可。

PHP 的时间日期函数很常用，但并不复杂。读者一般只需要掌握 UNIX 时间戳的获得操作方法格式化字符的使用方法，即可轻松掌握 PHP 时间日期的处理。下面程序 7-32.php 以制作简易日历的例子，作为时间日期函数的综合练习。

程序 7-32.php

```
<?php
 $year=@$_GET ['year'];
 $month=@$_GET ['month'];
 if(empty($year))$year=date("Y");
 if(empty($month))$month=date("n");
 $day=date("j");
 $wd_ar=array("日","一","二","三","四","五","六");
 $wd=date("w",mktime(0,0,0,$month ,1,$year ));
 $y_lnk1=$year <=1970?$year=1970:$year -1;
 $y_lnk2=$year>=2037?$year =2037:$year +1;
 $m_lnk1=$month<=1?$month =1:$month-1;
 $m_lnk2=$month>=12?$month =1:$month+1;

 echo"<table cellpadding=6 cellspacing=0 width=200 bgcolor=yellow>
 <tr align=center bgcolor=yellow>";
 echo"<td colspan=4><a href='7-32.php?year=$y_lnk1&month=&month'>
```

```
  <</a>".$year."年<a href='7-32.php?year=$y_lnk2&month=$month'>></a></td>";

echo"<td colspan=3><a href='7-32.php?year=$year&month=$m_lnk1'><</a>".$month .
"月<a href='7-32.php?year=$year&month=$m_lnk2'>></a></td></tr>";
echo"<tr align=center>";
for($i=0;$i<7;$i++)
{
    echo"<td>$wd_ar[$i]</td>";
}
echo"</tr>";
$tnum=$wd+date("t",mktime(0,0,0,$month,1,$year));
for($i=0;$i<$tnum;$i++)
{
  $date=$i+1-$wd;
  if($i%7==0)echo"<tr align=center>";
  echo"<td>";
  if($i>=$wd)
  {
     if($date ==$day&&$month==date("n"))
     echo"<font color=red><b>".$day."</font></b>";
     else
     echo $date;
  }
  echo"</td>";
  if($i%7==6)echo"</tr>";
}
echo"</table>";
?>
```

程序中第 2～第 3 行从地址栏分别获得年份和月份，第 4～第 5 行分别初始化本年度年份和本月的月份，第 6 行获取当天是本月的第几天。第 8 行计算当月第 1 天是星期几，第 09～第 12 行计算年月份点击后的变化，第 14～第 20 行输出日历头，第 21～第 27 行输出星期日到星期六的周日排列，第 28～第 44 行输出日历。其中，第 37 行将当月当天的日期用红色加粗显示。程序运行结果如图 7-29 所示。

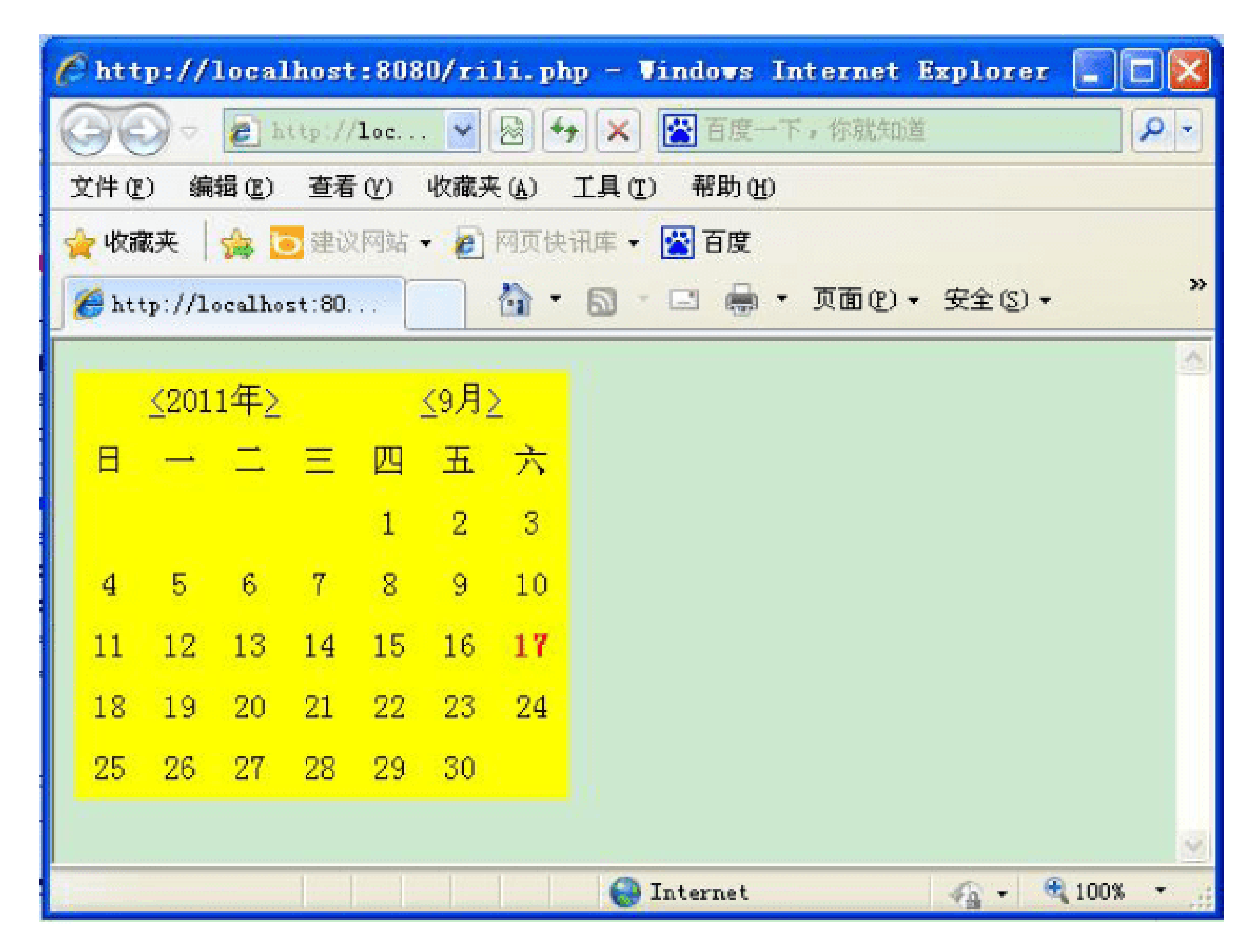


图 7-29 程序 7-32.php 的运行结果

7.6　正则表达式

通俗地讲，正则表达式（Regular Expression）是用一个“字符串”描述一个特征，然后去验证另一个“字符串”是否符合这个特征。例如，表达式“ab+”描述的特征是“一个‘a’和任意个‘b’，那么，‘ab’‘abb’‘abbbbbb’都符合这个特征。正则表达式是一种可以用于模式匹配和替换的强有力的工具，可以让用户通过使用一系列的特殊字符构建匹配模式，然后把匹配模式与数据文件、程序输入以及 Web 页面的表单输入等目标对象进行比较，根据比较对象中是否包含匹配模式，执行相应的程序。

正则表达式的用途非常广泛，如：

① 进行数据有效性检查。

② 用来查找字符串。

③ 用来替换，比普通的替换更强大。

PHP 中有两种形式的正则表达式，分别被称为“POSIX 扩展”正则表达式和“Perl 兼容”正则表达式。Perl 兼容正则表达式功能更强大，且是二进制安全的。POSIX 的正则函数库，自 PHP 5.3 以后，就不在推荐使用，从 PHP 6 以后，就将被移除，所以本节只介绍 Perl 兼容正则表达式。

7.6.1　正则表达式的组成元素

一般地，一个正则表达式由原子（字母、逗号、数值等）和元字符（星号、圆括号等）以及模式修正符三部分组成的文字模式所构成，其中元字符联合构成了一个文本模式的程序性描述，能允许编程人员匹配文本内的模式和子模式。模式是正则表达式最基本的元素，是一组描述字符串特征的字符。模式可以很简单，由普通的字符串组成，也可以非常复杂，往往用特殊的字符表示一个范围内的字符、重复出现，或表示上下文。例如，^once，这个模式包含一个特殊的字符^，表示该模式只匹配那些以 once 开头的字符串。例如，该模式与字符串"once upon a time"匹配，与"There once was a man from NewYork"不匹配。与 Perl 兼容的正则表达式函数中使用模式时，一定要给模式加上定界符，即将模式包含在两个反斜线“/”之间。

1. 原子

原子是构成正则表达式的基本单位，有以下 4 种：

① 普通字符：a～z、A～Z、0～9 和_。

② 单元符：每个单元符用圆括号括起来作为一个完整单位，如（abc)，（opm)。

③ 原子表：一组原子放在“[ ]”中，如［abcs]。

④ 转义字符。

\d 为匹配一个数字 0～9　[0～9]。

\D 为匹配除数字以外的任何一个字符 [^0～9]。

\w 为匹配一个英文字母，数字或下划线 [0～9、a～z、A～Z_]。

\W 为匹配除一个英文字母，数字或下划线外的任何一个字符[^0～9、a～z、A～Z_]。

\s 为匹配一个空白字符（包含如下字符，都可以匹配出来）[\f\n\r\t\v]。

\f 为匹配一个换页字符。

\n 为匹配一个换行字符。

\r 为匹配一个回车字符。

\t 为匹配一个制表符。

\v 为匹配一个垂直制表符。

\S 为匹配除空白符以外的任何一个字符　[^\f\n\r\t\v]。

\oNN 为匹配一个八进制数。

\xNN 为匹配一个十六进制数。

2. 元字符

用于构成正则表达式的具有特殊功能用途的字符。表 7-7 中列出了正则表达式中用到的元字符。

表 7-7　正则表达式中的元字符

元　字　符	说　　明
\	转义字符
|	匹配其中任一个
()	把字符、元字符和子表达式组合成组，合并整体匹配，并放入内存，可使用\1 \2…依次获取
[]	建立一个字符类，该类型将匹配方括号内所包含的任意一个字符
{}	为前导表达式定义一个最小/最大匹配数目
^	匹配字符串首部内容
$	匹配字符串尾部内容
*	匹配前一个内容的 0 次、1 次或多次
?	匹配前一个内容的 0 次或 1 次
+	匹配前一个内容的 1 次或多次
.	匹配内容的 0 次、1 次或多次，但不包含回车换行符
-	在一个字符类中指定一个字符的范围
\b	匹配单词边界，边界可以是空格或者特殊符合
\B	匹配除带单词边界意外内容

3. 模式修正符

模式修正符是为正则表达式增强和补充的一个功能，使用在正则之外，增强了正则表达式的处理能力。

运算顺序如下：

（1）遵循从左到右的运算规则。

（2）( )圆括号的优先级最高。

（3）* ? + {}　重复匹配内容，为第二优先级。

（4）^ $ \b　边界处理，为第三优先级。

（5）|条件处理符，为第四优先级。

最后按照运算顺序计算匹配。

表 7-8 中列出了正则表达式中用到的模式修饰符。

表 7-8　正则表达式中的模式修饰符

字　符	说　明
i	正则内容在匹配时候不区分大小写（默认是区分的）
m	在匹配首内容或者尾内容时候采用多行识别匹配
S	将转义回车取消是为单行匹配
x	忽略正则中的空白
A	强制从头开始匹配
D	强制$匹配尾部无任何内容 \n
U	禁止贪婪匹配，只跟踪到最近的一个匹配符并结束，常用在采集程序上的正则表达式

7.6.2　常用正则表达式

（1）非负整数：^\d+$。

（2）正整数：^[0-9]*[1-9][0-9]*$。

（3）非正整数：^((-\d+)|(0+))$。

（4）负整数：^-[0-9]*[1-9][0-9]*$。

（5）整数：^-?\d+$。

（6）非负浮点数：^\d+(\.\d+)?$。

（7）正浮点数：(0-9)+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*))$。

（8）非正浮点数：^((-\d+\.\d+)?)|(0+(\.0+)?))$。

（9）负浮点数：^(-((正浮点数正则式)))$。

（10）英文字符串：^[A-Za-z]+$。

（11）英文大写串：^[A-Z]+$。

（12）英文小写串：^[a-z]+”。

（13）英文字符数字串：^[A-Za-z0-9]+$。

（14）英数字加下划线串：^\w+$。

（15）E-mail 地址：^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$。

（16）URL：^[a-zA-Z]+://(\w+(-\w+)*)(\.(\w+(-\w+)*))*(\?\s*)?$。

7.6.3　正则表达式函数

1. 全局匹配函数

```
int preg_match_all(string $pattern, string $subject, array $matches[, int
$flags])
```

函数说明：在 subject 中搜索所有与 pattern 给出的正则表达式匹配的内容，并将结果

以 flags 指定的顺序放到数组 matches 中。搜索到第一个匹配项之后，接下来的搜索从上一个匹配项末尾开始。

程序 7-33.php

```
<!--程序 7-33.php: preg_match_all 函数 -->
<?php
$html_text = '<a href="http://www.cmbchina.com/" title="招商银行">招商银行</a><br>
<a href="http://www.abchina.com/cn/" title="农业银行">农业银行</a><br>
<a href="http://www.icbc.com.cn/icbc/" title="工商银行">工商银行</a><br>
<a href="http://cn.unionpay.com/" title="中国银联">中国银联</a><br>';
preg_match_all('/a href="(.*?)"/', $html_text, $href);
preg_match_all('/title="(.*?)"/', $html_text, $title);
print_r($href[1]);
echo "<br><br>";
print_r($title[1]);
?>
```

程序运行结果如图 7-30 所示。

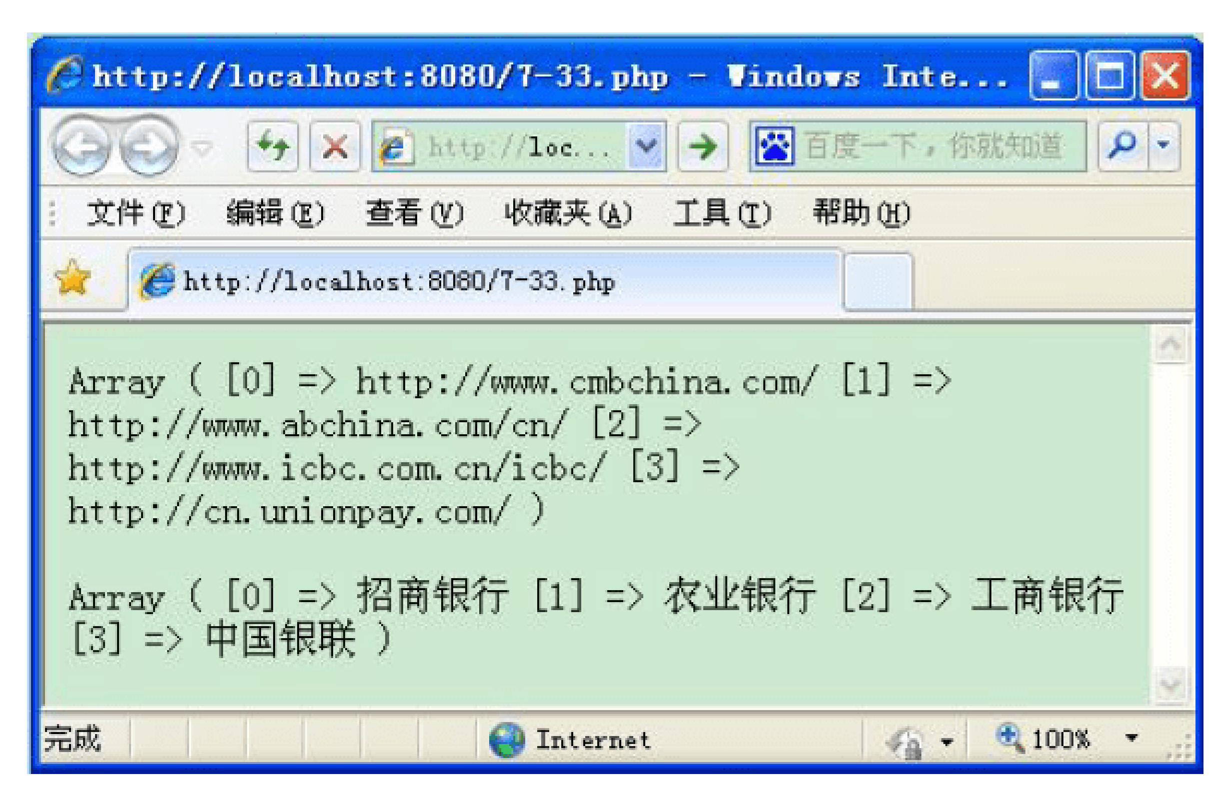


图 7-30 程序 7-33.php 的运行结果

2. 替换函数

```
mixed preg_replace(mixed $pattern, mixed $replacement, mixed $subject[, int $limit])
```

函数说明：在 subject 中搜索 pattern 模式的匹配项并替换为 replacement。如果指定了 limit，则仅替换 limit 个匹配，如果省略 limit 或其值为−1，则所有的匹配项都会被替换。

程序 7-34.php

```
<!--程序 7-34.php: preg_ replace 函数 -->
<?php
$html_text = '<a href="http://www.cmbchina.com/" title="招商银行">招商银行</a><br>
<a href="http://www.abchina.com/cn/" title="农业银行">农业银行</a><br>
<a href="http://www.icbc.com.cn/icbc/" title="工商银行">工商银行</a><br>
<a href="http://cn.unionpay.com/" title="中国银联">中国银联</a><br>';
echo preg_replace('/a href="(.*?)"/','a href=\\1 target="_blank"', $html_text );
```

```
    echo preg_replace('/a href="(.*?)"/', 'a href="\\1" target="_blank"',
$html_text,2 );
?>
```

程序运行结果如图 7-31 所示。

3. 正则切割函数

```
array preg_split(string $pattern, string $subject[, int $limit[, int $flags]])
```

函数说明：返回一个数组，包含 subject 中沿着与 pattern 匹配的边界所分割的子串。如果指定了 limit，则最多返回 limit 个子串，如果 limit 是–1，则意味着没有限制，可以用来继续指定可选参数 flags。

程序 7-35.php

```
<!--程序 7-35.php: preg_split 函数 -->
<?php
$html_text = '<a href="http://www.cmbchina.com/" title="招商银行">招商
银行 </a><br>
<a href="http://www.abchina.com/cn/" title="农业银行">农业银行</a><br>
<a href="http://www.icbc.com.cn/icbc/" title="工商银行">工商银行</a><br>
<a href="http://cn.unionpay.com/" title="中国银联">中国银联</a><br>';
print_r(preg_split('/a href="(.*?)"/',$html_text ));
?>
```

程序运行结果如图 7-32 所示。

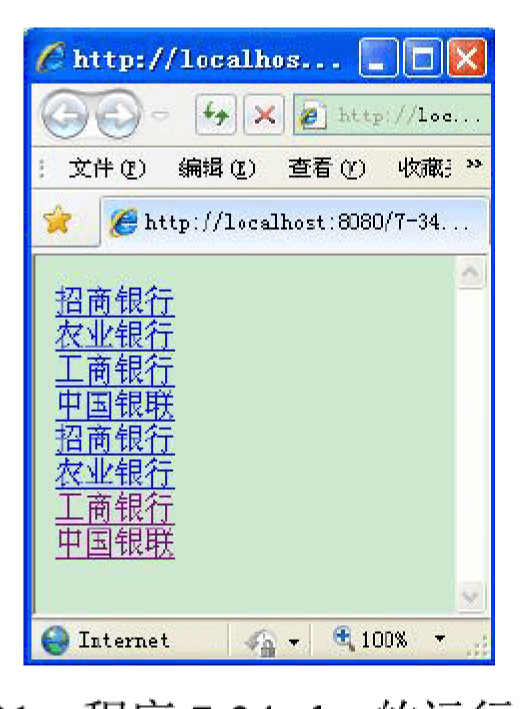


图 7-31　程序 7-34.php 的运行结果

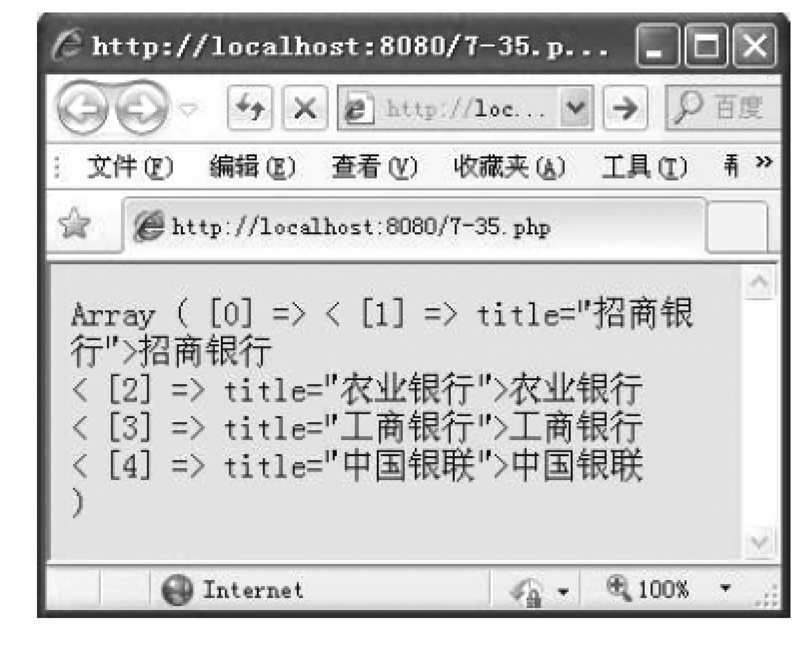


图 7-32　程序 7-35.php 的运行结果

4. preg_grep 函数

```
array preg_grep(string $pattern, array $input[, int $flags])
```

函数说明：返回一个包括了 input 数组中与给定的 pattern 模式相匹配单元的新数组。如果参数 flags 为 PREG_GREP_INVERT，则返回的是由不匹配 pattern 单元所组成的新数组。

程序 7-36.php

```
<!--程序 7-36.php: preg_grep 函数 -->
<?php
$arr = array("选修 1"=>"Data Base", "选修 2"=>"Data Mining", "选修
3"=>"DSP", "选修 4"=>"JAVA", "选修 5"=>"C#");
$newarr = preg_grep("/D+/", $arr);
print_r($newarr); //输出: Array ( [选修 1] => Data Base [选修 2] => Data
Mining [选修 3] => DSP )
```

```
$newarr = preg_grep("/D+/", $arr, PREG_GREP_INVERT);
echo"<br><br>";
print_r($newarr); //输出: Array ( [选修 4] => JAVA [选修 5] => C# )
?>
```

程序就运行结果如图 7-33 所示。

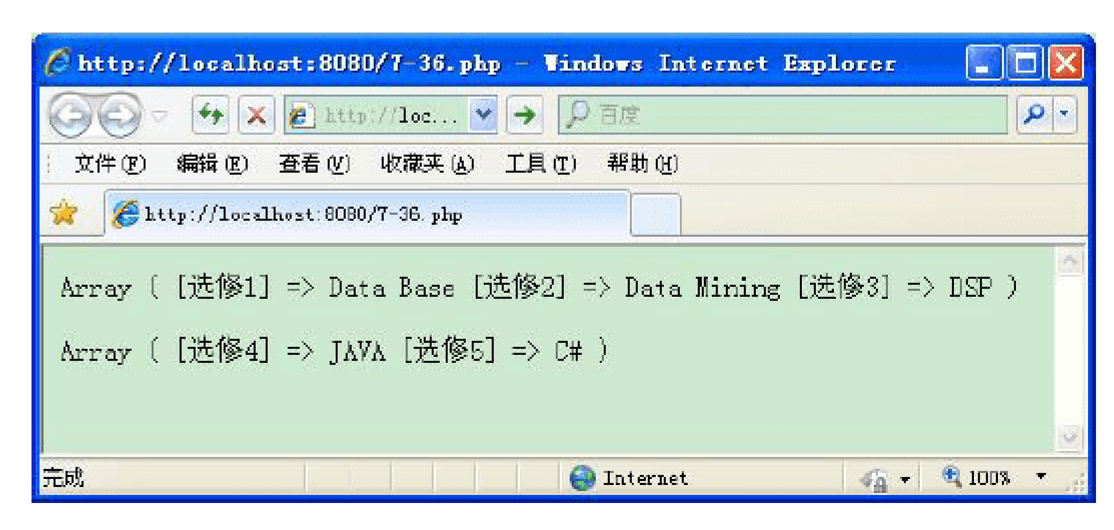


图 7-33　程序 7-36.php 的运行结果

5. preg_match 函数

```
int preg_match(string $pattern, string $subject[, array $matches [, int $flags]])
```

函数说明：在 subject 字符串中搜索与 pattern 给出的正则表达式相匹配的内容。函数应用的实例程序如 7-37.php 所示。

程序 7-37.php

```
<!--程序 7-37.php: preg_match 函数 -->
<?php
$html_text = '<a href="http://www.cmbchina.com/" title="招商银行">招商银行</a><br>
<a href="http://www.abchina.com/cn/"  title="农业银行">农业银行</a><br>
<a  href="http://www.icbc.com.cn/icbc/"  title="工商银行">工商银行</a><br>
<a href="http://cn.unionpay.com/" title="中国银联">中国银联</a><br>';
preg_match('/a href="(.*?)"/', $html_text, $href);
preg_match('/title="(.*?)"/', $html_text, $title);
print_r($href[1]);       //将输出 http://www.cmbchina.com/
echo"<br><br>";
print_r($title[1]);      //将输出“招商银行”
?>
```

程序运行结果如图 7-34 所示。

6. preg_quote 函数

```
string preg_quote(string $str[, string $delimiter])
```

函数说明：以 str 为参数并给其中每个属于正则表达式语法的字符前面加上一个反斜线。如果需要以动态生成的字符串作为模式去匹配，则可以用此函数转义其中可能包含的特殊字符。如果提供了可选参数 delimiter，该字符也将被转义。可以用来转义 PCRE 函数所需要的定界符，最常用的定界符是斜线 /。正则表达式的特殊字符包括 .\+*?[^]$(){}=!<>|:等。

程序 7-38.php

```
<!--程序 7-38.php: preg_quote 函数 -->
<?php
```

```
echo preg_quote('a href="(.*?)"'); //将输出 a href\="\(\.\*\?\)"
echo"<br><br>";
echo preg_quote('a href="(.*?)"', 'e'); //将输出 a hr\ef\="\(\.\*\?\)"
?>
```

程序运行结果如图 7-35 所示。



图 7-34　程序 7-37.php 的运行结果

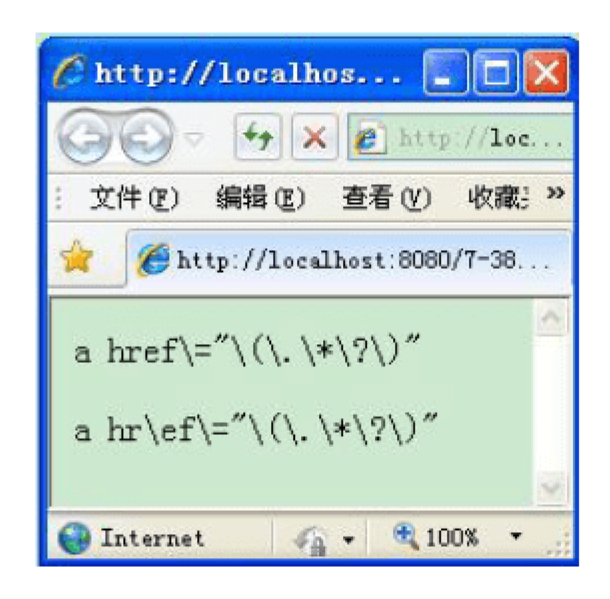


图 7-35　程序 7-38.php 的运行结果

7. preg_replace_callback 函数

```
mixed preg_replace_callback(mixed $pattern, callback $callback, mixed
$subject[, int $limit])
```

函数说明：本函数的行为几乎与 preg_replace()一样，除了不是提供一个 replacement 参数，而是指定一个 callback 函数。该函数将以目标字符串中的匹配数组作为输入参数，并返回用于替换的字符串。函数应用的实例程序如 7-39.php 所示。

程序 7-39.php

```
<!--程序 7-39.php: preg_replace_callback 函数 -->
<?php
$html_text = '<a href="http://www.cmbchina.com/" title="招商银行">招商
银行</a><br>
<a href="http://www.abchina.com/cn/" title="农业银行">农业银行</a><br>
<a href="http://www.icbc.com.cn/icbc/" title="工商银行">工商银行</a><br>
<a href="http://cn.unionpay.com/" title="中国银联">中国银联</a><br>';
function add_target($matches){
 return 'a href="'.$matches[1].'"target="_blank';
}
echo preg_replace_callback('/a href="(.*?)"/','add_target',$html_text);
?>
```

程序运行结果如图 7-36 所示。

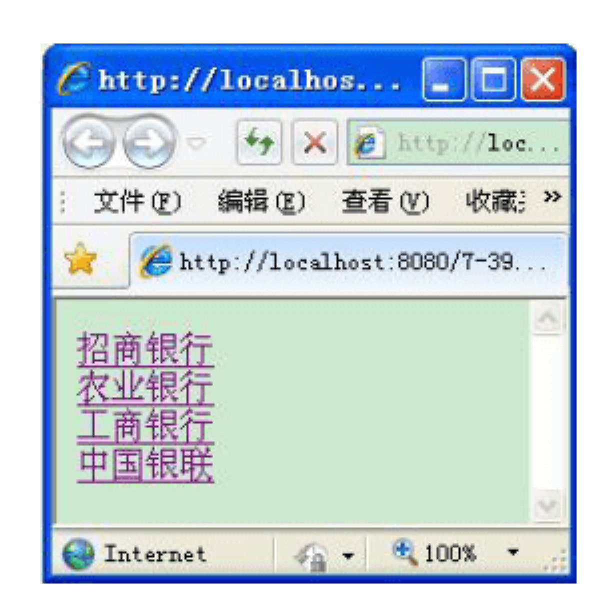


图 7-36　程序 7-39.php 的运行结果

7.6.4 正则表达式 Web 验证实例

在以下的例子中，综合应用了正则表达式在验证手机号码、电子信箱等方面的应用。

程序 7-40（1）.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 7-40(1).php：正则函数综合实例-->
<head>
<title>会员注册</title>
</head>
<body>
<form name="form" method="post" action="7-40.php">
<div align="center"><font size="4" >会员注册</font></div>
<table style="font-size: 13px"  align="center" border="1">
<tr><td>用户：</td>
<td><input type="text" name="user">
<td >* 不超过 10 个字符(数字，字母和下划线)</td></tr>
<tr><td>密码：</td>
<td><input type="password" name="psd" size="21"></td>
<td >* 6 到 14 个数字</td></tr>
<tr><td>手机：</td>
<td><input type="text" name="phone"></td>
<td >* 11 位数字，第 1 位为 1</td></tr>
<tr><td>邮箱：</td>
<td><input type="text" name="Email"></td>
<td >* 有效的邮件地址</td></tr>
<tr><td colspan="3" align="center">
<input type="submit" name="GO" value="注册">   
<input type="reset" name="NO" value="取消">
</td></tr>
</table>
</form>
</body>
</html>
```

以上是注册会员的注册页面，以下程序利用正则表达式对注册信息进行验证。

程序 7-40（2）.php

```
<!--程序 7-40(2).php：正则函数综合实例-->
<?php
include 'EX4_4_Hpage.php';                    //包含程序 EX4_4Hpage.php
$id=$_POST['user'];
$pwd=$_POST['psd'];
$phone=$_POST['phone'];
$Email=$_POST['Email'];
$checkuser=preg_match('/^\w{1,10}$/',$id);  //检查字符串是否在10个字符以内
$checkpsd=preg_match('/^\d{6,14}$/',$pwd);  //检查是否在 6～14 个字符之间
$checkphone=preg_match('/^1\d{10}$/',$phone);//检查是否是以1开头的11位数字
```

```
//检查 Email 地址的合法性
$checkEmail=preg_match('/^[a-zA-Z0-9_\-]+@[a-zA-Z0-9\-]+\.
 [a-zA-Z0-9\-\.]+$/',$Email);
 if($checkuser&&$checkpsd&&$checkphone&&$checkEmail) //如果都为 1，则注册成功
   echo "注册成功！";
else
   echo "注册失败，格式不对";
?>
```

当用户输入表单上要求的各项，全部符合所列条件时，则输出“注册成功!”；否则，输出“注册失败!”。实例运行结果如图 7-37 所示。



图 7-37　程序 7-40.php 的运行结果

7.7　文件处理

在网络编程中要用到的文件操作大致可以分为两大类，一种是普通文件的操作；另一种是数据库文件的操作。在普通文件的操作中对记事本文件的操作最为简单，下面就来探讨一下 PHP 对文件（以记事本为例）的操作。PHP 提供了一些文件操作的函数，常用函数如表 7-9 所示。

表 7-9　PHP 提供的文件操作函数

函　数　名	功　　能	函　数　名	功　　能
basename	返回路径中的文件名部分	feof	测试文件指针是否到了文件结束的位置
chmod	改变文件模式	fgetc	从文件指针中读取字符
clearstatcache	清除文件状态缓存	fgets	从文件指针中读取一行
delete	参见 unlink()或 unset()	file_exists	检查文件或目录是否存在
disk_free_space	返回目录中的可用空间	file_put_contents	将一个字符串写入文件
diskfreespace	disk_free_space()的别名	fileatime	取得文件的上次访问时间

续表

函　数　名	功　　能	函　数　名	功　　能
filegroup	取得文件的组	flock	轻便的咨询文件锁定
filemtime	取得文件修改时间	fopen	打开文件或者 URL
fileperms	取得文件的权限	fputcsv	将行格式化为 CSV 并写入文件指针
filetype	取得文件类型	fread	读取文件（可安全用于二进制文件）
fnmatch	用模式匹配文件名	fseek	在文件指针中定位
fpassthru	输出文件指针处的所有剩余数据	ftell	返回文件指针读/写的位置
fputs	fwrite()的别名	fwrite	写入文件（可安全用于二进制文件）
fscanf	从文件中格式化输入	is_dir	判断给定文件名是否是一个目录
fstat	通过已打开的文件指针取得文件信息	is_file	判断给定文件名是否为一个正常的文件
ftruncate	将文件截断到给定的长度	is_readable	判断给定文件名是否可读
glob	寻找与模式匹配的文件路径	is_uploaded_file	判断文件是否是通过 HTTP POST 上传的
is_executable	判断给定文件名是否可执行	is_writeable	is_writable()的别名
is_link	判断给定文件名是否为一个符号连接	linkinfo	获取一个连接的信息
chgrp	改变文件所属的组	mkdir	新建目录
chown	改变文件的所有者	parse_ini_file	解析一个配置文件
copy	复制文件	pclose	关闭进程文件指针
dirname	返回路径中的目录部分	readfile	输出一个文件
disk_total_space	返回一个目录的磁盘总大小	realpath	返回规范化的绝对路径名
fclose	关闭一个已打开的文件指针	rewind	倒回文件指针的位置
fflush	将缓冲内容输出到文件	set_file_buffer	stream_set_write_buffer()的别名
fgetcsv	从文件指针中读入一行并解析 CSV 字段	symlink	建立符号连接
fgetss	从文件指针中读取一行并过滤掉 HTML 标记	tmpfile	建立一个临时文件
file_get_contents	将整个文件读入一个字符串	umask	改变当前的 umask
file	把整个文件读入一个数组中	is_writable	判断给定的文件是否可写
filectime	取得文件的 inode 修改时间	link	建立一个硬连接

续表

函　数　名	功　　能	函　数　名	功　　能
fileinode	取得文件的 inode	lstat	给出一个文件或符号连接的信息
fileowner	取得文件的所有者	move_uploaded_file	将上传的文件移动到新位置
filesize	取得文件大小	pathinfo	返回文件路径的信息
popen	打开进程文件指针	stat	给出文件的信息
readlink	返回符号连接指向的目标	tempnam	建立一个具有唯一文件名的文件
rename	重命名一个文件或目录	touch	设定文件的访问和修改时间
rmdir	删除目录	unlink	删除文件

有关详细功能和使用方法请参见 PHP 5 的帮助文档。

7.7.1　文件的打开与读写

要想利用 PHP 对文件进行操作，就要先了解有关 PHP 中打开和读写文件的相关函数。

1. fopen 函数

fopen 函数格式如下：

```
resource fopen (string filename, string mode [, bool use_include_path ])
```

fopen 函数的作用是打开文件或 URL。其中，filename 是要打开的文件名，必须为字符串形式。如果 filename 是“scheme://…”（如 http://…）的格式，则被当成一个 URL，PHP 将搜索协议处理器（也被称为封装协议）来处理此模式。如果 PHP 认为 filename 指定的是一个本地文件（如 num.txt），将尝试在该文件上打开一个流。该文件必须是 PHP 可以访问的，因此需要确认文件访问权限允许该访问。mode 是打开文件的方式，必须为字符形式，可以取以下几个值。

'r'：只读形式，文件指针指向文件的开头。

'r+'：可读可写，文件指针指向文件的开头。

'w'：只写形式，文件指针指向文件的开头，打开同时清除所有内容，如果文件不存在，将尝试建立文件。

'w+'：可读可写，文件指针指向文件的开头，打开同时清除所有内容，如果文件不存在，将尝试建立文件。

'a'：追加形式（只可写入），文件指针指向文件的最后，如果文件不存在，将尝试建立文件。

'a+'：可读可写，文件指针指向文件的最后，如果文件不存在，将尝试建立文件。

2. fgets 函数

fgets 函数的格式如下：

```
string fgets (int handle [, int length])
```

fgets 函数的功能是从文件指针中读取一行。其中，handle 是要读入数据的文件流指针，

由 fopen 函数返回数值。length 是要读入的字符个数，实际读入的字符个数是 length−1。

从 handle 指向的文件中读取一行并返回长度最多为 length−1 字节的字符串。碰到换行符（包括在返回值中）、EOF 或已经读取了 length−1 字节后停止（看先碰到哪一种情况）。如果没有指定 length，则默认为 1KB，也就是说 1024 字节。出错时返回 FALSE。

3. fwrite 函数

fwrite 函数格式如下：

```
int fwrite (resource handle, string string [, int length])
```

fwrite 函数的功能是写入文件，与 int fputs(resource handle, string str, int [length])功能相同。

fwrite()把 string 的内容写入文件指针 handle 处。如果指定了 length，当写入了 length 个字节或者写完了 string 以后，写入就会停止。

fwrite()返回写入的字符数，出现错误时返回 FALSE。

4. fclose 函数

fclose 函数的格式如下：

```
bool fclose (resource handle)
```

fclose 函数的功能是关闭一个已打开的文件指针，即将 handle 指向的文件关闭。如果成功则返回 TRUE，失败则返回 FALSE。

文件指针必须有效，并且是通过 fopen()或 fsockopen()成功打开的。

下面就用以上几个简单的文件操作函数来编写一个统计征订教材的应用程序。程序由两个文件组成：程序 7-41.php jczd.html 以表单的形式允许用户填写征订的教材的相关信息；程序 7-42jczdaction.php 处理程序 7-41.php jczd.html 程序提交的信息，并作出对用户的响应。

程序 7-41.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head> <!--文件 7-41jczd.html:文件处理综合实例-->
<meta http-equiv="Content-Type" content="text/html; charset="gb2312" />
<title>教材征订表</title>
</head>
<body>
<h3 align="center"><font size="5" color="#FF9900">教材征订表 </font>
</h3>
<form id="form1" name="form1" method="post" action="jczdaction.php">
<table width="90%" border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td align="center">订书人</td>
    <td align="center">教材名称</td>
    <td align="center">书号</td>
    <td align="center">出版社</td>
    <td align="center">编著者 </td>
    <td align="center">单价 </td>
```

```
    <td align="center">使用年级及专业 </td>
  </tr>
  <tr>
    <td><input align="center" name="dsr" type="text" size="10" /></td>
    <td><input align="center" name="jcname" type="text" size="30" /></td>
    <td><input align="center" name="ISBN" type="text" size="12"/></td>
    <td><input align="center" name="press" type="text" size="7"/></td>
    <td><input align="center" name="author" type="text" size="10"/></td>
    <td><input align="center" name="price" type="text" size="10"/></td>
    <td><input align="center" name="grade" type="text" size="18"/></td>
  </tr>
<tr><td colspan="7" align="center"><input name="Sumit" type="submit"
value="提交" />   
  <input name="Reset" type="reset" value="重置" /></td>
</tr>
</table><p> </p>
</form>
</body>
</html>
```

程序运行结果如图 7-38 所示。



图 7-38　程序 7-41 jczd.html 的运行结果

程序 7-41 jczd.html 提交的数据被下面的程序 7-42jczdacution.php 接收并处理。

程序 7-42jczdaction.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head> <!--文件 7-42jczdaction.php:文件处理综合实例-->
<meta http-equiv="Content-Type" content="text/html; charset="utf-8" />
<title>教材征订反馈信息</title>
</head>
<body>
<?php
$dsr=$_POST ["dsr"];
$jcname=$_POST ["jcname"];
$isbn=$_POST ["ISBN"];
```

```
$press=$_POST ["press"];
$author=$_POST ["author"];
$price=$_POST ["price"];
$grade=$_POST ["grade"];
////////////////////////////////////////////////////
if($dsr=="")echo"<font color=red>订书人</font>不能为空!"."<br>";
if ($jcname =="") echo"<font color=red>教材名称</font>不能为空!"."<br>";
if($isbn=="")echo"<font color=red>书号</font>不能为空!"."<br>";
if($press=="")echo "<font color=red>出版社</font>不能为空!"."<br>";
if($author=="")echo"<font color=red>编著者</font>不能为空!"."<br>";
if($price=="")echo"<font color=red>定价</font>不能为空!"."<br>";
if($grade=="")echo"<font color=red>使用年级及专业</font>不能为空!"."<br>";
if($dsr==""||$jcname ==""||$isbn ==""||$press ==""||$author ==""||$press
==""||
$grade==""){echo"<font color=blue>请返回重新填写!</font>";;}
else{
  ///////////////////////////////////////////////////
 if (!file_exists("jczd.txt")){     //如果文件不存在
   $fp=fopen("jczd.txt", "w");      //借助 w 参数，创建文件
   fclose($fp);                     //关闭文件
  // echo "num.txt 文件创建成功! <br>";
  }
  $fp=fopen("jczd.txt","r");
  @$num=fgets($fp,12);              //读取 11 位数字
  if ($num=="") $num=0;
   //如果文件的内容为空，初始化为 0
  $num++;                           //浏览次数加一
  @fclose($fp);                     //关闭文件
  $fp=fopen("jczd.txt", "w");       //只写方式打开 num.txt 文件
  fwrite($fp,$num);                 //写入加一后结果
  fclose($fp);                      //关闭文件
/////////////////////////////////////////
 if (!file_exists("jczd.xls"))
 {                                  //如果文件不存在
  $fp=fopen("jczd.xls", "w");       //借助 w 参数，创建文件
$tablehead="序号"."\t"."订书人"."\t"."教材名称"."\t"."书号"."\t"."出版社
"."\t".
    "编著者"."\t"."定价"."\t"."使用年级及专业";
  fwrite($fp,$tablehead);
  fclose($fp);    //关闭文件
  }
  $result="\n".$num."\t".$dsr."\t".$jcname."\t".$isbn."
  \t".$press."\t".$author."\t".$price."\t".$grade;
  $fp=fopen("jczd.xls", "a");       //只写方式打开 num.txt 文件
  fwrite($fp,$result);
  fclose($fp);                      //关闭文件
/////////////////////////////////////////////////////////
echo"你是"."<font color=red>".$dsr."<br>"."</font>"."你输入的内容是:"  ;
echo"<table width=900 border=1 aglign=center >" ;
echo"<tr >"."<font color=blue>";
```

```
59  echo"<td>"."订书人 ". "</td>" ;
60  echo"<td>"."教材名称". "</td>". "</td>"."<td>"."书号 "."</td>".
61  "<td>"."出版社"."</td>"."<td>"."编著者"."</td>"."<td>"."定价"."</td>".
62  "<td>"."使用年级及专业"."</td>";
63  echo"</tr>"."</font>";
64  echo"<tr>"."<font color=green>";
65  echo"<td>".$dsr. "</td>"."<td>".$jcname. "</td>"."<td>"
    .$isbn."</td>".
66  "<td>".$press."</td>"."<td>".$author."</td>"."<td>".$price."</td>".
67  "<td>".$grade."</td>";
68  echo"</br>";
69  echo"</table>";
70    }
71  ?>
72  </body>
73  </html>
```

程序第 9～第 15 行接收 jczd.html 上传的数据，第 17～第 25 行检查上传数据的格式，在上传数据合乎要求的前提下，第 28～第 41 行创建 jczd.txt 文件，文件中保存了数据提交的次数。第 43～第 54 行创建文件 jczd.xls，文件以电子表格的形式存储了提交的信息。程序在当前目录下创建的文件如图 7-39 所示。

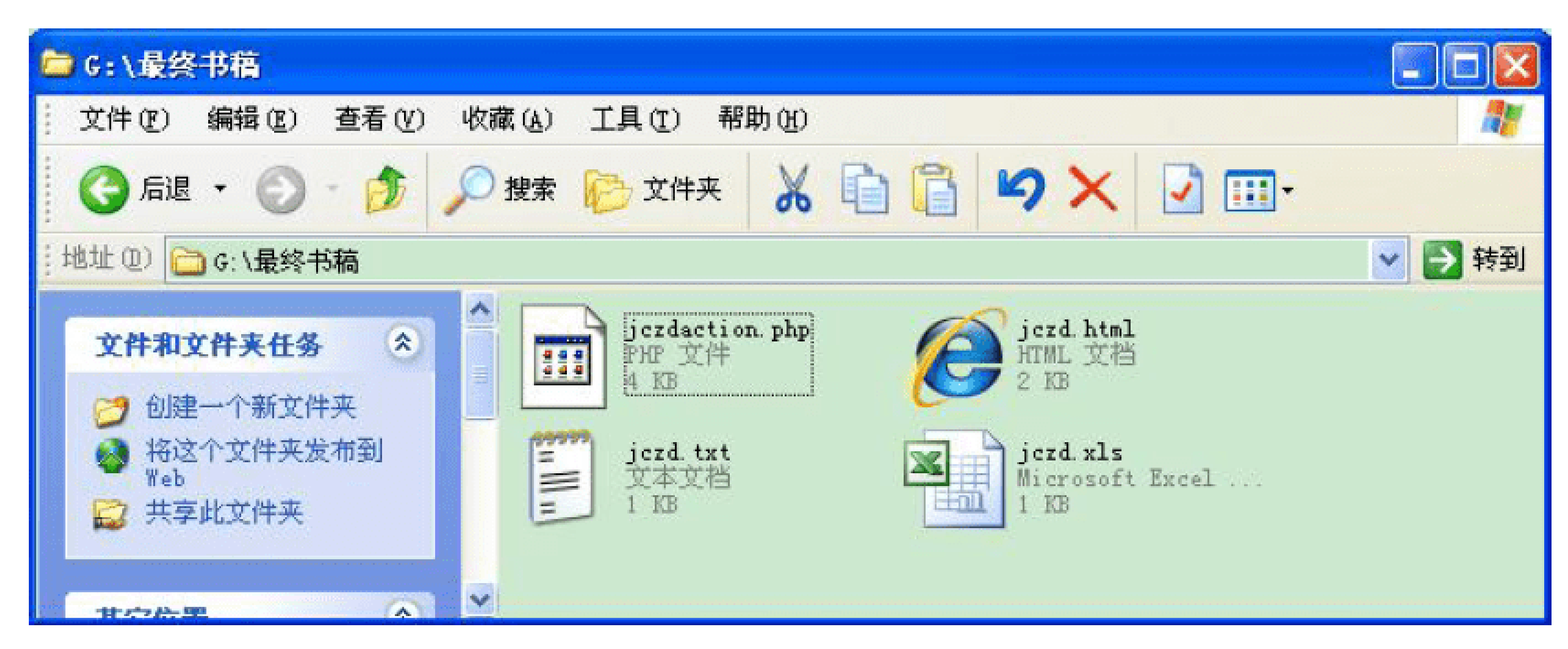


图 7-39　程序 7-42.php 创建的文件

程序运行结果如图 7-40 所示。

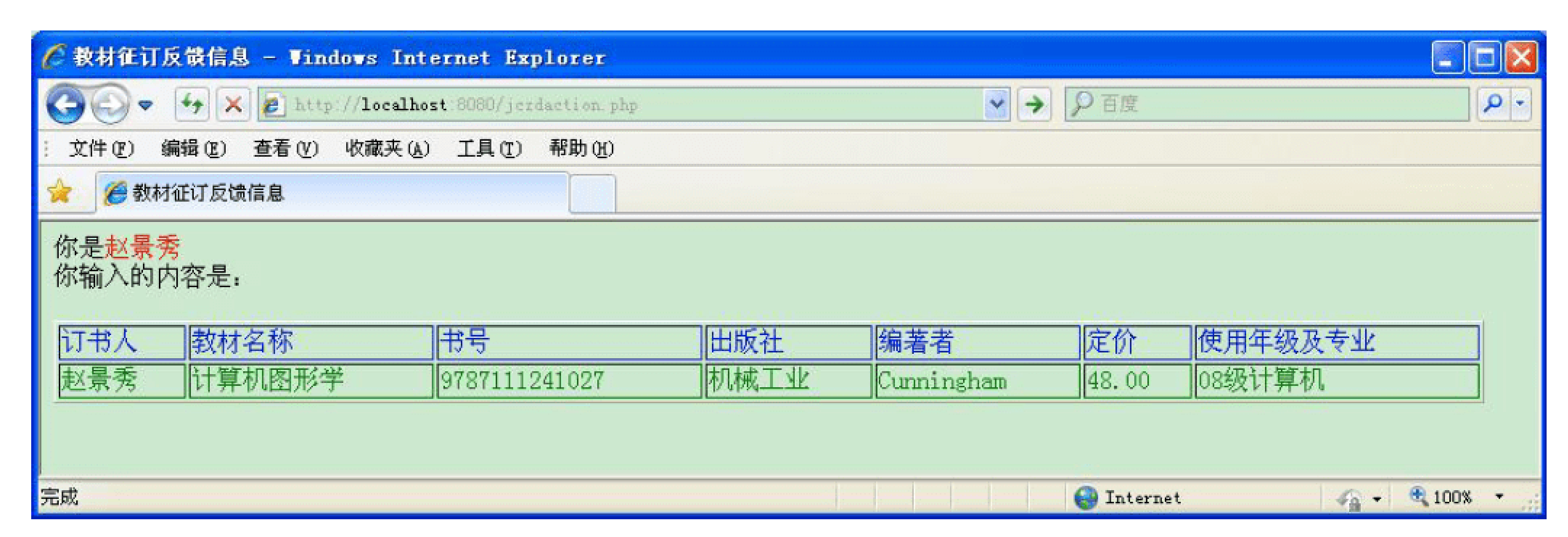


图 7-40　程序 7-42.php 的运行结果

7.7.2 目录的创建、删除与遍历

目录的操作主要是利用相关的目录函数实现的，先来看一下与目录操作有关的函数。

1. string getcwd (void)

功能：取得当前工作目录。

2. bool chdir (string directory)

功能：将当前目录改为 directory。

3. new dir(sting directory)

功能：将输入的目录名转换为一个对象并返回。例如：

```
class dir
{
    dir(string directory )
    string path
    resource handle
    string read ( void )
    void rewind ( void )
    void close ( void )
}
```

该对象含有 3 个属性和 3 个方法。

2 个属性为：

- handle：目录句柄。
- path：打开目录的路径。

3 个方法为：

- read (void)：读取目录。
- rewind (void)：复位目录。
- close (void)：关闭目录。

这 3 个方法与后面将要介绍的 readdir()、rewinddir()、closedir()函数的作用相同。

4. resource opendir (string path)

功能：打开目录句柄。path 为要打开的目录路径。

5. string readdir (resource dir_handle)

功能：返回目录中下一个文件的文件名。文件名以在文件系统中的排序返回。

6. dir_handle

dir_handle 为目录句柄的 resource，之前由 opendir()打开。

功能：成功则返回文件名，失败返回 FALSE。

7. void rewinddir (resource dir_handle)

功能：倒回目录句柄。将 dir_handle 指定的目录流重置到目录的开头。dir_handle 为目录句柄的 resource，之前由 opendir()打开。

8. void closedir (resource dir_handle)

功能：关闭目录句柄。关闭由 dir_handle 指定的目录流。流必须已被 opendir()打开。

9. array scandir (string directory [, int sorting_order])

功能：列出指定路径中的文件和目录。返回一个 array，包含有 directory 中的文件和目录。

参数 directory 是要被浏览的目录。

参数 sorting_order 是文件的排列顺序，默认的排序顺序是按字母升序排列。如果使用了可选参数 sorting_order（设为 1），则排序顺序是按字母降序排列。

10. bool chroot (string directory)

将当前进程的根目录改变为 directory。

本函数仅在系统支持且运行于 CLI、CGI 或嵌入 SAPI 版本时才能正确工作。此外本函数还需要 root 权限。现举例如下：

程序 7-43.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 7-43.php：目录输出-->
<head>
<title>目录输出</title>
</head>
<body>
<?php
  $dir=getcwd();                          //获取当前路径
  echo getcwd(). "<br>";                  //输出当前目录
  $files1=scandir($dir);                  //列出指定路径中的文件和目录
  $files2=scandir($dir,1);
  print_r($files1);                       //输出指定路径中的文件和目录
  echo"<br><br>";
  print_r($files2);
  echo"<br>";
  $dir=dir($dir);
  echo gettype($dir)."<br>";
  echo "目录句柄:".$dir->handle."<br>";
  echo "目录路径:".$dir->path."<br>";
  while ($entry=$dir->read())
   echo $entry."<br>";
  $dir->close();
  if (chdir("c:/windows")){
   echo "当前目录更改成功:c:/windows<br>";
    }else{
   echo "当前目录更改失败! <br>";
    }
?>
</body>
</html>
```

程序 7-43.php 运行结果如图 7-41 所示。

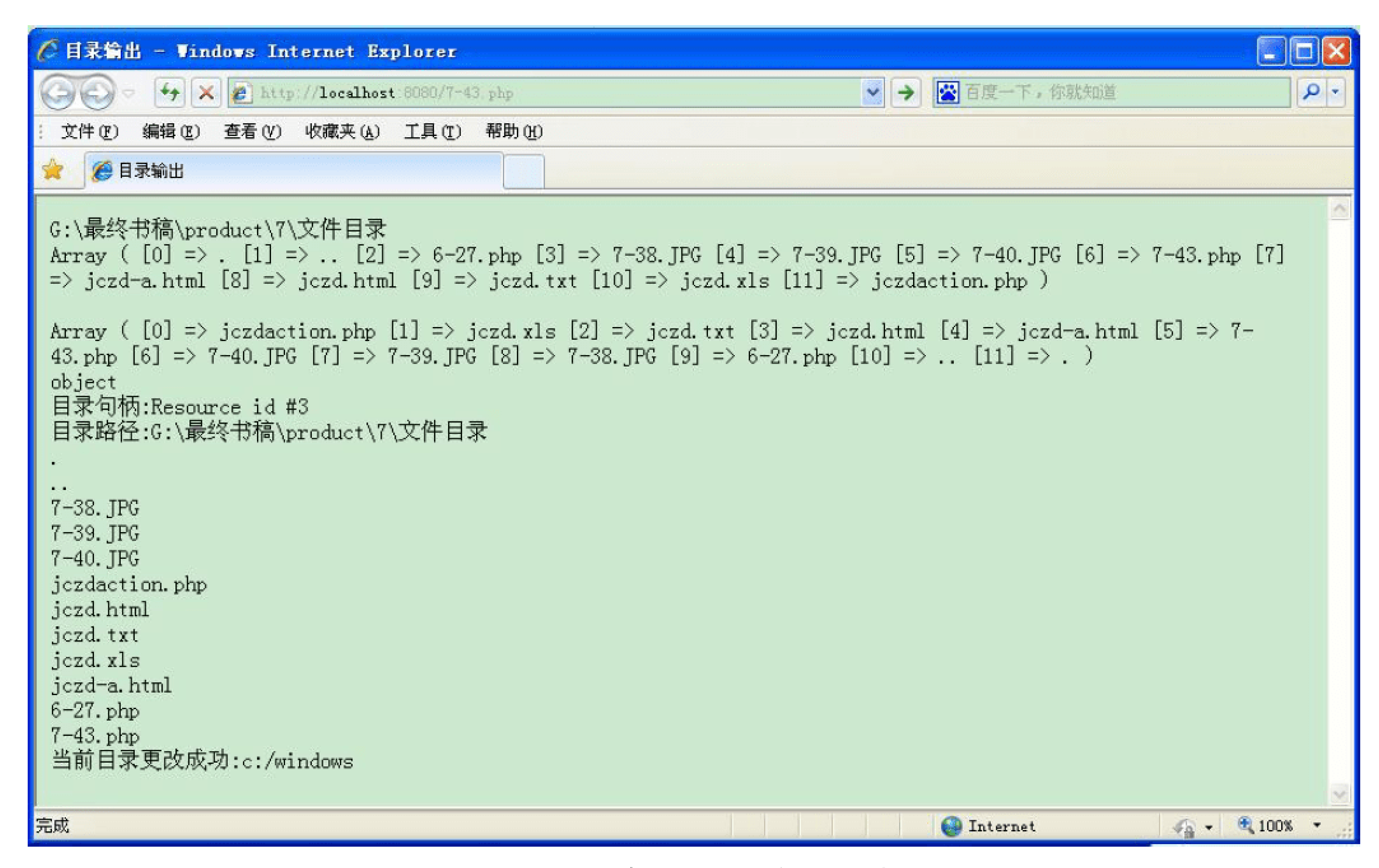


图 7-41　程序 7-43.php 的运行结果

下面再来看一个例子，体会目录操作在实际中的应用。

程序 7-44.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 7-44.php: 遍历图片显示-->
<head>
    <title>遍历图片显示</title>
</head>
<body>
  <?php
   $addr=getcwd();
   $dir=dir($addr);
   while ($file_name=$dir->read()){
   if ($file_name=="." or $file_name==".."){
     }else{
     echo "<img src=".$addr.$file_name." width=40 height=30>  ";
     }
   }
  ?>
</body>
</html>
```

其运行结果如图 7-42 所示。



图 7-42　程序 7-44.php 的运行结果

7.8　本章小节

本章在 PHP 基本语法的基础上，详细介绍了 PHP 中的数组处理，字符串操作，图形图像的绘制，时间日期的运用，正则表达式的验证和文件读写、目录的操作。PHP 的简单易用很大程度上依赖于其提供的这些功能。本章涉及的函数众多，语法烦琐，功能齐全，是进行 PHP 开发必不可少的内容。读者务必多加练习，熟练掌握。

7.9　练习题

1．怎样创建一个一维数组并输出数组信息？
2．遍历数组的方法有几种？各自有怎样用法？
3．哪个函数能计算字符串的长度？怎样比较字符串？
4．求子串的函数是哪个？需要什么样的参数？
5．哪个函数具有子串替换功能？需要什么样的参数？
6．绘制一幅图像需要几个步骤？定制背景颜色和前景颜色的函数是同一个吗？
7．绘制图像时用哪个函数指定图像的类型？
8．什么是 UNIX 时间戳？UNIX 纪元是什么时间？
9．用什么函数将时间戳转化为时间？
10．什么是正则表达式？它有几种形式？
11．写出验证电子邮件的正则表达式。
12．如何创建一个文件？操作文件的步骤有哪些？
13．编程实现向一个已经存在的文件写入追加的内容。
14．编程实现读取一个页面的内容并显示。
15．编程实现文件的上传与下载功能，上传的压缩文件能自动解压到指定目录。

第8章 PHP 5面向对象编程

本章重点

- 类的概念;
- 访问控制;
- 构造器和方法重载;
- 类的继承;
- 接口。

8.1 PHP面向对象概述

面向对象程序设计（Object-Oriented Programming，OOP）是近代程序设计领域的一大革命。它提高了程序设计者的产能，提高了软件的重复使用率，并且降低维护成本。面向对象程序设计将要解决的实际问题抽象成程序中的一个个对象，并赋予对象特殊的性质，这样可以克服许多传统的面向过程的编程语言（或称程序性语言）的许多缺点，是一种更新、更先进的编程方式，也是目前已经大规模使用的编程思想。

PHP 支持面向对象的编程方法。尤其是到了 PHP 5，面向对象的特性被大大加强，虽然目前 PHP 作为一门 Web 开发语言，其对面向对象特性的支持程度与纯粹的面向对象语言如 Java 等还有差距，但在一定程度上已经给 PHP 开发者带来很大便利。因此掌握 PHP 面向对象编程就显得十分有必要。

关于面向对象的编程方法，是一门专门的学问，有很多著作阐述这个问题。关于类、对象、方法、成员、继承、封装、重载、构造器等名词，对于一个没有接触过面向对象编程的读者来说可能是一头雾水。由于阐述面向对象的原理不是本书的主要内容，因此本书不去深入探讨这些内容，但是会在遇到每个名词时进行尽量简短、通俗的解释。如果读者在此之前就具备了面向对象编程的经验，那么学习本章将会非常轻松。如果读者没有任何面向对象编程知识，也能够通过下面的介绍慢慢领会。建议这类读者先参阅一些专门介绍面向对象编程的书籍，这样再学习起来会更加顺畅。

必须说明的是：PHP 支持面向对象的编程，但并不是只能以面向对象方式编程。换句话说，不了解 PHP 的面向对象编程并不影响 PHP 的学习。即使本章完全跳过，也基本不

会影响后续章节的学习。事实上，由于 Web 程序规模有限，每个网页程序一般是几十行到几百行，数千甚至数万行的程序并不多见。再加上 Web 程序基本上是独立的个体，它们运行时互不相干，这就使得单个网页程序的内部逻辑比较简单，并不是十分复杂，因此面向对象的编程思路并发挥不出太大作用。于是实际开发中大多数 PHP 开发者主要采用传统的“面向过程”的方法。大多数人只是部分使用面向对象的方法甚至完全不使用。因此，即使读者对面向对象的编程没有经验，学习本章感到困难，也不必丧失信心。面向对象到目前为止还没有成为 PHP 的核心。

8.2　类与对象

面向对象的程序设计就是把程序分为各个部分，每个部分完成不同的任务，这些部分就是对象。同简单的函数或一组变量相比，对象的功能十分丰富。因为对象把函数和变量以易于使用的方式封装在一起，同时又把其内部的实现细节隐藏于其他对象。

如何创建对象呢？用类！类与对象的关系，就像模具与铸件的关系。现实世界中的任何事物都是对象，任何对象都可归属于某类事物。类就是对具有一定相同属性的对象的抽象。在程序设计中，先定义类，然后再用类创建对象。下面先看一个描述几何图形圆的类（程序 8-1.php）。

程序 8-1.php

```
<!--程序 8-1.php：描述圆的类-->
<?php
class Circle{
 protected $radius;
 public function __construct($radius){
   $this->radius=$radius;
   echo"构造函数执行了！"."<br>";
  }
 public function Perimeter(){
   return 2*3.14*$this->radius;
  }
 public function Area (){
   return 3.14*$this->radius*$this->radius;
  }
 public function __destruct(){
   echo "析构函数执行了!";
  }
 }
$myC1=new Circle(1.0);
echo $myC1->Perimeter()."<br>";
echo$myC1->Area();
 ?>
```

与其他面向对象的语言一样，PHP 也是通过 class 关键字声明类的开始的，class 后面的是类名 Circle。Circle 中定义了一个受保护的变量$radius 来存放圆的半径；定义了两个公有方法，分别用于计算圆的周长和面积。注意，PHP 中类内变量是以“$this->变量名”

的方式访问的。另外，Circle 中还定义了构造方法。PHP 中任何类的构造方法名都是_ _construct()。不同于 C++和 Java，PHP 中要访问类的成员时，要用“$this->成员名”的方式进行访问。程序第 16 行使用 new 运算符创建了一个 Circle 的对象并将其命名为$myC1。第 17、18 两行分别用“对象名→”的方式访问了对象的成员方法。

程序的第 15 行定义了一个析构函数。PHP 中的析构函数名字规定为_ _destruct()，它不能带有参数，也不要在程序中显式地调用一个对象的析构函数。PHP 将在对象被销毁前调用析构函数将对象从内存中销毁。

PHP 中成员的访问控制有 3 种，分别是 private、protected 和 public。



图 8-1　类示例

private 声明为私有的属性和方法，只可以在类的内部访问，私有的属性和方法不能被继承。

protected 声明为保护的属性和方法，可在类的内部和子类中访问。

public 声明为公有的属性和方法，在类的内部和外部都能访问，是成员默认的访问控制属性。 程序运行结果如图 8-1 所示。

8.3　构造函数与析构函数

构造函数（构造器）是面向对象编程中的一种重要机制，本节介绍 PHP 中的构造函数与析构函数。

8.3.1　构造函数

所谓构造函数，是指类中的一个特殊的方法，在 PHP 中，这个方法以_ _construct()为方法名，而且此方法没有返回值。

构造函数的实质是类中的一个方法，也就是类的一个成员。只不过这个方法和其他方法相比有其特殊性。其特殊性主要表现在以下几个方面：

（1）构造函数必须命名为_ _construct()。在 PHP 5 以前版本中构造函数名必须与类名一致。PHP 5 以后改为_ _construct()（注意 construct 前面是连续两条下划线“_”）。

（2）构造函数在类被实例化时自动调用（用 new 创建对象时自动运行此方法）。

（3）构造函数没有返回值。

（4）构造函数一般不被“显式”调用。也就是说构造函数在创建对象时自动运行，而不需要人为去调用。

构造函数作为类的基本特性之一，必然有其用途。构造函数最主要的用途就是用来实现类的初始化。即创建一个对象的同时就对这个对象进行一些初始化操作，而不是等到对象创建完成后再逐个设置，这时构造函数就派上了用场。

下面来看一个例子。

程序 8-2.php

```
<!--程序 8-2.php：构造函数的使用-->
```

```
02  <?php
03  class student{
04    private $name;
05    private $age;
06    private $id;
07   function  __construct($xm,$nl,$xh){              //构造函数
08    $this->name = $xm;
09    $this->age  = $nl;
10    $this->id   = $xh;
11    echo "<p>构造函数已经被执行! </p>";
12     }
13   function  toString(){                         //普通方法
14    $info  = "姓名: ".$this->name."<br>";
15    $info .= "年龄: ".$this->age."<br>";
16    $info .= "学号: ".$this->id;
17    return $info;
18    }
19    }
20   $stu1 = new student("刘太强",22,"201101020122");
21   echo $stu1->toString();
22   ?>
```

在程序 8-2.php 中，类 student 中定义了构造函数，构造函数接收 3 个参数并分别赋值给类的 3 个成员变量。本程序中第 20 行创建了一个 student 类的对象$stu1，在用 new 关键字创建对象的同时，提供 3 个初始化参数值，这样 3 个值被传递到构造函数_ _construct()中，完成赋值。因此第 21 行直接输出，可以看到提供的初始信息已经成功保存到对象之中。这个例子足以看到构造函数所起的作用。

程序 8-2.php 的运行结果如图 8-2 所示。

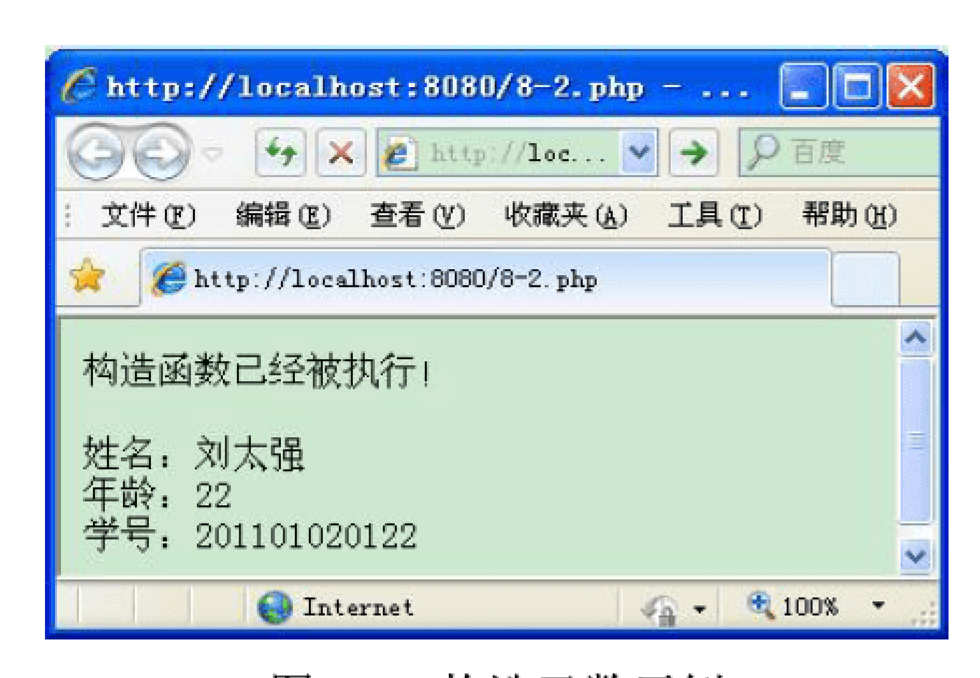


图 8-2　构造函数示例

其实“构造函数”仍然是一个方法。在面向对象编程中，已经基本废弃了“函数”这个称呼，类中的函数称为“方法”。之所以仍然将构造器称为“构造函数”，只是一种习惯。在许多资料中都这么称呼，本书仍然沿用这一称呼。

8.3.2　析构函数

析构函数是与构造函数相对应的另一个特殊方法。析构函数是类中的一个方法，不过

有其特殊性。

（1）析构函数必须命名为_ _destruct()。

（2）析构函数没有参数。

（3）析构函数没有返回值。

（4）析构函数在对象被销毁时自动调用，一般不需要显式调用。

析构函数是 PHP 5 中引入的新概念。

学习了构造函数之后，析构函数便很容易理解了，下面直接来看一个例子。

程序 8-3.php

```
<!--程序 8-3.php：析构函数的使用-->
<?php
 class MyDestructableClass {
  private $name;
  function __construct() {
   echo "构造函数已执行<br>";
   $this->name = "MyDestructableClass";
    }
  function __destruct() {
   echo "析构函数已执行（" . $this->name . "）";
    }
  }
 $obj = new MyDestructableClass();
?>
```

程序 8-7.php 运行结果如图 8-3 所示。



图 8-3 析构造函数示例

这说明在对象创建时，构造函数执行。当脚本执行完毕，销毁对象时，析构函数被执行（一个 PHP 脚本的所有语句执行完毕即说明脚本执行完成，此时服务器自动销毁对象，释放资源）。

8.4 类的继承

继承（Inheritance）是面向对象程序设计的三大特性之一。所谓继承，就是指一些类可以从另外一些已有的类中派生而来。如类 B 派生自类 A，则称 A 为“父类”或“超类”，B 为“子类”或“次类”。

继承是代码重用，提高编码效率的重要手段。PHP 中允许通过继承其他类的方法调用

这些类里已经定义好的属性和方法，但 PHP 只支持单继承，不支持多继承。要实现继承，只需要在类定义时使用 extends 关键字，即可让该类继承自另外一个类，从而拥有它所继承的类的全部特征。下面看一个示例。

程序 8-4.php

```
<!--程序 8-4.php：类的继承-->
<?php
class Circle{
 protected $radius;
 public function __construct($radius){
   $this->radius=$radius;
   echo"构造函数执行了！"."<br>";
  }
 public function Perimeter(){
   return 2*3.14*$this->radius;
  }
 public function Area (){
   return 3.14*$this->radius*$this->radius;
   }
 public function toString(){
  echo "圆的面积是:".$this->Area()."<br>";
  }
}

class Cylinder extends Circle{
  private $height;
 public function __construct($radius,$height){
  $this->height=$height;
  $this->radius=$radius;
    }
 public function BasalArea(){
  return parent::Area();
    }
 public function Area(){
  return 2*parent::Area()+2*3.14*$this->radius*$this->height;
  }
 public function Volume(){
  return   3.14*$this->radius*$this->radius*$this->height;
  }
 public function toString(){
  echo "圆柱体的底面积是：".$this->BasalArea()."<br>";
  echo "圆柱体的表面积是：".$this->Area()."<br>";
  echo "圆柱体的体积是：".$this->Volume();
      }
}
 $obj=new Cylinder(1.0,1.0);
 $obj->toString();
?>
```

程序 8-4.php 运行结果如图 8-4 所示。



图 8-4　类的继承

下面来分析一下本程序。

第 3～第 16 行定义了一个 Circle 类。

第 18～第 34 行定义了一个 Cylinder 类。在类声明中使用 extends 关键字，声明本类是从 Circle 类继承而来，那么 Cylinder 类就是 Circle 类的子类。Circle 类是父类。

第 35 行创建了一个 Cylinder 类的对象$obj。

第 36～第 40 行分别调用$obj 对象的 3 个方法。

虽然这个程序并不复杂，但是足以表明“继承”的作用。在类 Cylinder 中，并没有声明描述半径的变量，因为它从 Circle 中继承了描述半径的变量$radius，所以在 Cylinder 的构造方法中可以直接给它赋值。继承过程中，如果子类中有自己的构造方法，执行自己的构造方法；否则，子类在实例化时会自动调用父类的构造方法。

类 Cylinder 中定义了和类 Circle 中同名的方法 Area。当子类中有和父类同名的成员时，子类中的成员会覆盖父类中的成员。子类中如果要访问父类中的同名成员，需要在使用关键字 parent，如程序第 25、29 行所示。

一旦使用 extends 关键字声明一个类 B 继承自另外一个类 A，那么类 B 就继承了类的所有成员（父类中声明为 private 的除外）。其中，包括类 A 从它的父类中继承下来的成员。这就是继承的核心特点。掌握了继承的概念，就可以在很多场合灵活运用，来解决编程中的实际问题。

最后需要说明的是，并不是所有的类都可以被其他类继承。如果一个类不希望被其他类继承，可以在声明此类时在前面增加 final 关键字。

```
final class BaseClass {                          //此类声明为 final 最终类
   public function test() {
      echo "just a test";
   }
}
class ChildClass extends BaseClass          //报错，因为声明为 final 的类不能被继承
{
}
```

重载（Overloading）是面向对象的编程语言 3 大特性之一“多态”的重要表现形式。重载可以用一句话简单概括：允许在同一个类中出现同名的变量或方法。

实际上纯粹的面向对象编程语言，都对重载有良好的支持。PHP 作为一门 Web 编程语言，对重载的支持并不理想。甚至可以说 PHP 根本不支持真正的重载。因为，PHP 不允许

一个类中出现两个同名的变量或者同名的方法；否则会报错。但是 PHP 通过几个所谓的 Magic Methods（魔法方法）实现了变相的重载。在 PHP 手册关于 PHP 5 面向对象编程的介绍中，就有重载一节。但是很明显这种重载方法只是低层次的。

虽然有一些拐弯抹角的方式可以实现一定意义上的重载，但由于 PHP 在重载方面并不成熟，在这里不再进行讨论。

8.5　抽象类和类的接口

PHP 5 不支持多重继承，而是用接口作为多重继承的替代。接口是一种特殊的抽象类。在面向对象领域，抽象类主要是用来进行类型隐藏，是对一系列看上去不同，但本质上相同的具体概念的抽象。如果一个类中没有足够的信息描绘一个具体的对象，这样的类就是抽象类。

抽象类使用关键字 abstract 定义，不能被实例化。一个抽象类中至少要有一个抽象方法，抽象方法也用关键字 abstract 定义。抽象方法只提供了方法的声明，不提供方法的具体实现。在抽象类中可以有普通的方法和属性，但只要有一个方法是抽象方法，这个类就必须声明为抽象类。下面程序 8-5 的例子定义了一个抽象类。

程序 8-5.php

```
<?php
//<!--程序 8-5.php：抽象类示例-->
abstract class Graph{
 protected $Red;
 protected $Green;
 protected $Blue;
 protected $BgRed;
 protected $BgGreen;
 protected $BgBlue;
public function setBgColor($R,$G,$B){
  $this->BgRed=$R;
  $this->BgGreen=$G;
  $this->BgBlue=$B;
 }
public function setColor($R,$G,$B){
  $this->Red=$R;
  $this->Green=$G;
  $this->Blue=$B;
 }
abstract function Draw();
    }
class Circle extends Graph{
  public $cx;
  public $cy;
  public $radius;
public function __construct($cx,$cy,$radius,$radius){
  $this->cx=$cx;
  $this->cy=$cy;
  $this->radius=$radius;
```

```
        }
public function Draw(){
  $image=imagecreate(500,300);
  //创建背景图形,背景色设为白色
  $background=imagecolorallocate($image,$this->BgRed,$this->
  BgGreen,$this->BgBlue);
  //定义画笔颜色
  $Color=imagecolorallocate($image,$this->Red,$this->Green,$this->Blue);
  //画一个椭圆
imageellipse($image,$this->cx,$this->cy,$this->radius,$this->radius,
$Color);
  header("Content-type: image/gif");        //发送头信息
  imagegif($image);                          //输出图形
  imagedestroy($image);                      //清除资源
    }
      }
    $obj=new Circle(230,150,200,200);
    $obj->setBgColor(220,255,255);
    $obj->setColor(2,2,220);
    $obj->Draw();
  ?>
```

程序运行结果如图 8-5 所示。注意，程序中使用了 header 函数，所以程序中不能使用任何空行，且第一行前也不能有任何空行或空格。

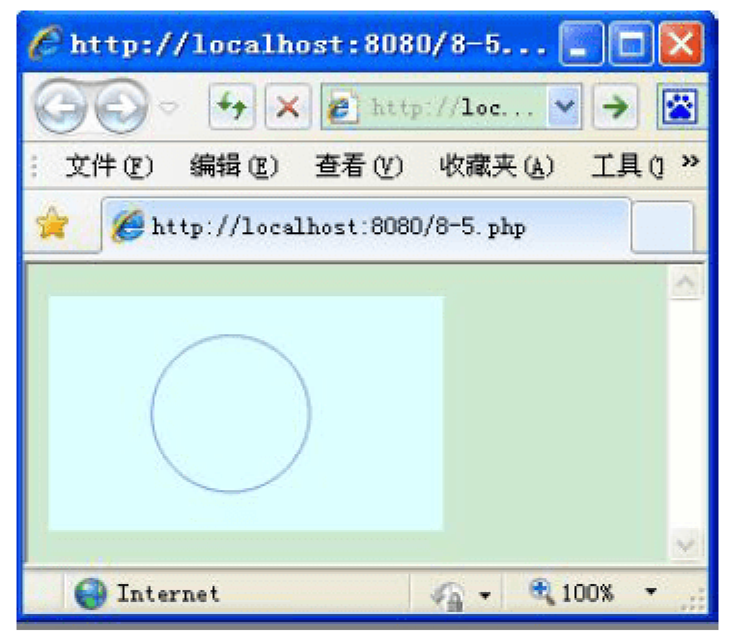


图 8-5 抽象类示例

抽象类用于描述抽象概念，其中声明的抽象方法为多个子类约定方法的声明，每个子类可根据自身的实际情况，给出抽象方法的具体实现，不同的子类可以有不同的实现，使得抽象类的多个子类能够表现出共同的行为。

抽象类中的方法，除了必须有至少一个抽象方法外，还可以有普通的方法。如果抽象类中没有普通方法，即全是抽象方法，这时的抽象类就成了抽象类中特殊的一种，称为接口。接口是一种特殊的抽象类，其中的方法都是抽象方法，属性也只能是用关键字 const 限定的常量。接口使用关键字 interface 定义。定义了接口之后可以将其实例化，接口的实例化称为接口的实现。接口的实现需要一个子类实现接口的所有抽象方法，该子类用 implements 关键字定义。接口的使用解决了多继承的问题。以下程序 8-6.php 中定义了两个接口，并在两个不同的类中分别得以实现。

程序 8-6.php

```
<!--程序 8-6.php: 接口示例-->
<?php
 interface Area{
 function area();
}
interface Volume{
 function Volume();
}
```

```
09   class Circle implements Area{
10    protected $radius;
11   public function __construct($r){
12    $this->radius=$r;
13    echo"构造函数执行了！"."<br>";
14    }
15   public function Area (){
16    return 3.14*$this->radius*$this->radius;
17   }
18   public function Perimeter(){
19    return 2*3.14*$this->radius;
20   }
21   public function toString(){
22    echo "圆的半径是:".$this->Area()."<br>";
23    echo "圆的周长是:".$this->Perimeter()."<br>";
24   }
25 }
26 class Globle extends Circle implements Area,Volume{
27     public function Volume(){
28     return 4/3*3.14*$this->radius*$this->radius*$this->radius;
29   }
30   public function toString(){
31    echo"球的半径是:".$this->radius."<br>";
32    echo"球的表面积是:".$this->Area()."<br>";
33    echo"球的体积是:".$this->Volume()."<br>";
34    }
35  }
36  $obj=new Circle(2.0);
37       $obj->toString();
38  echo "<br>";
39  $obj=new Globle(1,0);
40  $obj->toString();
41  ?>
```

程序的第 1～第 4 和第 6～第 8 行分别定义了两个接口 Area 和 Volume。第 10 行定义了类 Circle，实现了 Area 接口；第 28 行定义的类 Globle 继承了类 Circle 并实现了 Area 和 Volume 两个接口。程序运行结果如图 8-6 所示。

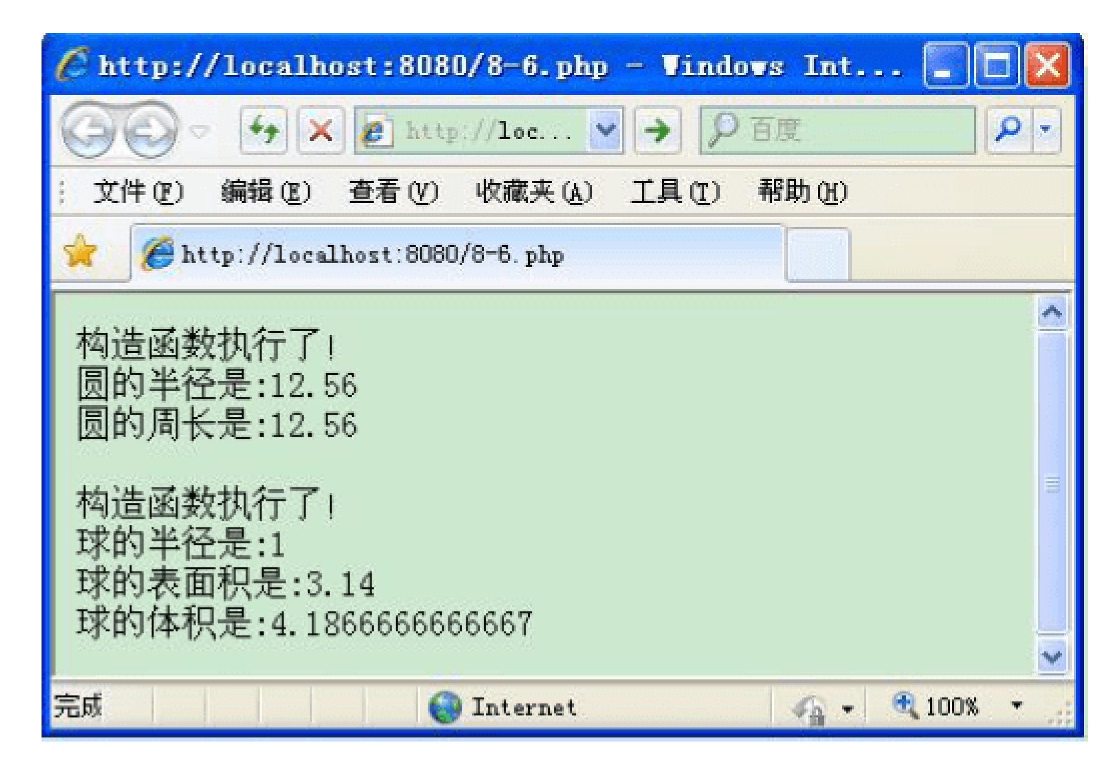


图 8-6　接口示例

8.6 错误和异常

在程序开发中，错误在所难免。有的是程序员的失误造成的，有的是程序运行环境造成的。如何去避免、调试、修复错误以及对程序可能发生的异常的处理是一个程序员必须具备的基本能力。

8.6.1 PHP 的错误处理

1. PHP 的错误

错误无非以下几种：

（1）语法错误：在程序中使用了错误的语法而导致的错误，是最为常见的错误。

（2）语义错误：在程序中正确地使用了 PHP 的语法，但是没有任何意义，程序达不到预想的效果。

（3）逻辑错误：在程序中使用的逻辑与实际上需要的逻辑不符。

（4）注释错误：在程序中写的注释与该程序代码的意义不符。

（5）运行时错误：由于运行环境等原因而导致的错误，与程序代码无关。PHP 程序解释器检测不到这类错误，只有在程序运行中才会发生。

不管是程序引发的错误，还是环境因素引发的错误，在默认情况下，PHP 都会给出提示信息。这些提示信息包含服务器的运行环境信息。在实际的 Web 环境中，将这些信息显示出来，必然给服务器带来安全隐患。因此，必须对可能出现的错误进行相应的处理。PHP 中的错误是通过一个错误级别进行划分的。共有 4 个级别：语法错误、致命错误、警告和通知。语法错误是在 PHP 解释器对 PHP 代码进行解析时产生的，致命错误是在运行 PHP 代码的过程中遇到环境或资源不可用导致的，这时脚本会终止执行。警告是代码运行过程中遇到的一些异常，如文件不存在或数据库打不开等。通知一般用于一些较小的错误，如使用没有赋值的变量等。警告和通知不会影响代码的整体运行。

2. 错误的处理

打开 php.ini 文件，里面有两项关于错误处理的设置：一个是 display_errors；另一个是 error_reporting。前一变量是用来告诉 PHP 是否显示错误，它的默认值为 Off，也即不显示错误信息，如果设置为 true，将显示错误信息；后一变量是告知 PHP 如何显示提示信息，默认值为 E_ALL，即显示所有的错误信息。

程序开发阶段，应打开错误报告，以方便调试，找到错误原因。一旦交付使用，就要关闭 display_errors 选项，以保障系统安全。程序运行中发生错误，可根据错误报告定位代码中的错误位置，检查存在的问题，或输出变量的中间结果，分析错误原因。调试错误最重要的是要有耐心，积累经验，养成良好的变成习惯。

8.6.2 PHP 的异常处理

在实际的系统运行中，有可能存在一些不可预知的错误，如文件权限不对、数据库无

法连接等。使用 PHP 5 以后版本提供的的异常处理方法，可更好地解决因环境等因素而引发的异常。

PHP 的异常处理机制类似 Java 的异常处理机制，也是用其关键字 try、catch 和 throw 来实现的。将需要进行异常处理的代码放入 try 代码块内，以便捕获可能存在的异常。每一个 try 语句必须至少有一个 catch 语句与之对应，catch 语句用于捕获异常。使用多个 catch 语句可以捕获不同的类所产生的异常。throw 语句用于抛出异常。PHP 中提供的异常类 Exception 定义如下：

```
<?php
 class Exception
 {
    // 属性
    protected $message ='Unknown exception';   // 异常信息
    protected $code = 0;                       // 用户自定义异常代码
    protected $file;                           // 发生异常的文件名
    protected $line;                           // 发生异常的代码行号

     // 方法
    final function getMessage();               // 返回异常信息
    final function getCode();                  // 返回异常代码
    final function getFile();                  // 返回发生异常的文件名
    final function getLine();                  // 返回发生异常的代码行号
    final function getTrace();                 // backtrace() 数组
final function getTraceAsString();   // 已格成化成字符串的 getTrace() 信息
 }
 ?>
```

可以扩展这个类，使之符合一定的个性化需要。

```
<?php
class myException extends Exception{
function __construct($message=null,$code=0)
{
parent::__construct($message,$code);
}
function __toString()
{
return '<div class="error">Exception'.$this->getCode().':'.
$this->getMessage().'in File:'.$this->getFile().'on line:'.$this-> getLine().
'</div>';                                      //改写抛出异常结果
}
}
```

下面的程序 8-7.php 演示了用这个类处理异常的用法。

程序 8-7.php

```
<!--程序 8-7.php: PHP 异常处理-->
<?php
```

```
class myException extends Exception{
function __construct($message=null,$code=0)
{
parent::__construct($message,$code);
}
function __toString()
{
return '<div class="error">Exception'.$this->getCode().':'
.$this->getMessage().
'in File:'.$this->getFile().' on line:'.$this->getLine().'</div>';
}
}
//以下为测试上面的类 myException 的部分
class TestException
{
   public $var;
   const THROW_NONE    = 0;
   const THROW_CUSTOM  = 1;
   const THROW_DEFAULT = 2;

function __construct($avalue = self::THROW_NONE) {
 switch ($avalue) {
   case self::THROW_CUSTOM:
      throw new MyException('1 is an invalid parameter', 5);
      break;

   case self::THROW_DEFAULT:
      throw new Exception('2 isnt allowed as a parameter', 6);
      break;

   default:
      $this->var = $avalue;
      break;
  }
 }
}
try {
 $o = new TestException(TestException::THROW_CUSTOM);
}catch (MyException $e) {       // 捕获异常
  echo "Caught my exception\n", $e;
  $e->customFunction();
} catch (Exception $e) {        // 被忽略
  echo "Caught Default Exception\n", $e;
}
?>
```

程序运行结果如图 8-7 所示。

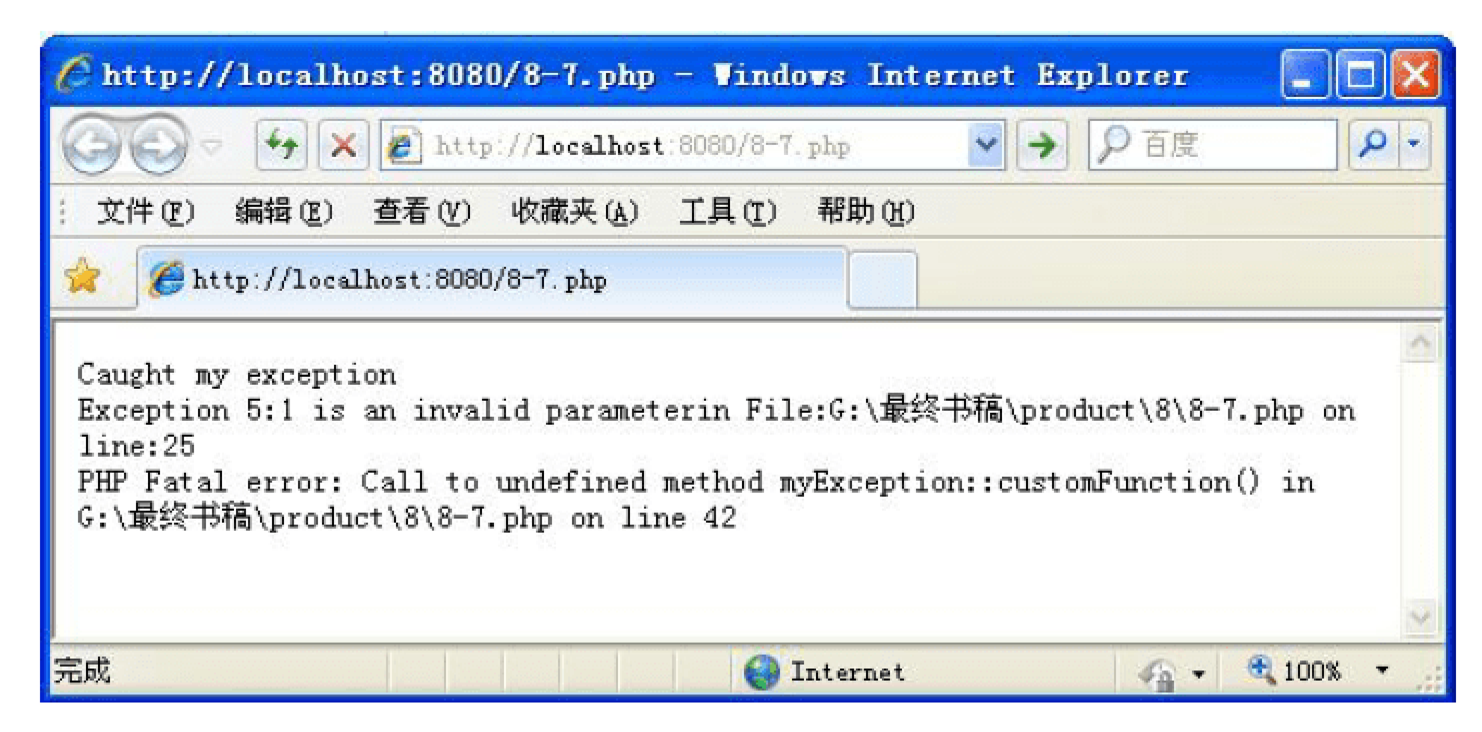


图 8-7　异常处理示例

8.7　本章小结

本章主要介绍了 PHP 的面向对象编程方法，主要包括类、对象、方法、属性的概念；类的定义；类的实例化；类的继承；覆盖和重载；抽象类和接口，以及用面向对象的方式处理程序错误和异常的方法。通过本章的学习，读者应当对 PHP 中面向对象编程部分有一个比较全面的了解，并初步掌握使用面向对象的编程方法编写 PHP 程序。

8.8　练习题

1．什么是面向对象的程序设计？

2．什么是类？PHP 中怎样定义一个类？

3．什么是构造函数？其作用是什么？

4．什么是析构函数？其作用是什么？

5．什么是类的继承？继承关系建立后，子类和父类之间函数以及属性有何变化？

6．什么是抽象类？什么是抽象方法？

7．什么是接口？怎样实现一个接口？

8．什么是异常？PHP 中怎样处理异常？

9．完善 Circle 类，增加处理异常的功能。

10．编写一个父类 Person（人），包含 name（姓名）、xb（性别）两个变量和一个构造器。在构造器中完成姓名、性别的初始化。编写一个子类 student（学生）继承自 Person，包含 xh（学号）变量和构造器。并在子类中定义一个方法，用来输出全部学生信息。

第9章 MySQL 数据库系统

本章重点

- MySQL 数据库的安装;
- 结构化查询语言（SQL）;
- MySQL 中的用户管理;
- MySQL 的数据管理;
- phpMyadmin 的使用。

9.1 MySQL 数据库简介

9.1.1 为什么选择 MySQL

动态网站开发离不开数据存储，数据存储则离不开数据库。目前应用中流行的后台数据库有 MySQL、SQL Server、Oracle、Sybase、DB2、PostgreSQL、Informix 等。PHP 支持几乎全部当前主流的数据库。但是，PHP 和 MySQL 的搭配无论从性能上还是到易用性上都毫无疑问地成为了开发者的首选。此外，还有一个重要原因就是 PHP 和 MySQL 都是免费和开放源代码的，并且都有良好的跨平台特性。这使得搭建 Web 服务器的成本几乎为零，而且开发出来的程序具有可移植性，这些都是吸引开发者的重要原因。

MySQL 是当今“世界上最流行的开源数据库”。权威调查机构 Evans 数据公司预测，相比其他的开源数据库和闭源数据库，越来越多的开发者将继续选择 MySQL。Evans 的总裁 John Andrews 表示，用户对 MySQL 和其他开源数据库的评价正在赶上甚至超过很多专有商业数据库软件。由于 MySQL 数据库已经如此普及，对企业来说它无疑是一个更好的选择。业界普遍的声音认为:“MySQL 是一个可靠的数据库系统，无论是在嵌入式或大型群集系统的部署中，还是在基于 Web 的应用程序领域，用户时常会发现其实自己并不是第一个选用 MySQL 数据库的先驱者。”大概是由于 PHP 开发者特别衷情于 MySQL，因此在 PHP 中建立了完美的 MySQL 支持。在 PHP 中，用来操作 MySQL 的函数一直是 PHP 的标准内置函数。开发者只需要用 PHP 写下短短几行代码，就可以轻松连接到 MySQL 数据库。PHP 还提供了大量的函数来对 MySQL 数据库进行操作。可以说，用 PHP 操作 MySQL 数据库极为简

单和高效，这也使得 PHP＋MySQL 成为当今最为流行的 Web 开发语言与数据库搭配之一。

9.1.2　MySQL 数据库简介

MySQL 是 MySQL AB 公司开的一种开放源代码的关系型数据库管理系统（RDBMS），MySQL 数据库系统使用最常用的数据库管理语言——结构化查询语言（SQL）进行数据库管理。由于 MySQL 是开放源代码的，因此任何人都可以在通用公共许可证（General Public License）的许可下下载并根据个性化的需要对其进行修改。MySQL 因为其速度、可靠性和适应性而备受关注。MySQL 关系型数据库于 1998 年 1 月发行第一个版本。它使用系统核心提供的多线程机制提供完全的多线程运行模式，提供了面向 C、C++、Eiffel、Java、Perl、PHP、Python 等编程语言的编程接口，支持多种字段类型并提供了完整的操作符。

2001 年 MySQL 4.0 版本发布，标志着 MySQL 已经长大成人。在这个版本中提供了许多新的特性，如新的表定义文件格式、高性能的数据复制功能、更加强大的全文搜索功能等。到 MySQL 5.1 版本，开始支持存储过程、触发器和视图等功能，性能和效率方面都得到了更大的提升，能满足企业级用户的需求。目前 MySQL 的最新版本是 5.5.17。现在的 MySQL 已经追赶上来了，相对于庞大的 SQL Server 它以优越的性能，灵活的易用性和跨平台等特点正在得到更多用户的认可。上海爱可生信息技术有限公司对 SQL Server 和 MySQL 的性能进行了比较，如表 9-1 所示。

表 9-1　MySQL 与 SQL Server 的性能比较

功　　能	MySQL	SQL Server
开源	√	
跨平台	√	
插件式存储引擎	√	
高可用集群数据库	√	√
ANSI SQL，子查询	√	√
存储过程，触发器，UDF	√	√
可更新视图	√	√
事务能力	√	√
分布式事务	√	√
行级锁	√	√
索引（clustered, b-tree, hash, full-text）	√	√
查询缓存	√	
Unicode，UTF-8 支持	√	√
XML,XPATH 支持	√	√
复制	√	√
表和索引的分区	√	√
高速数据加载	√	√
在线备份及时间点恢复	√	√

续表

功　　能	MySQL	SQL Server
表压缩及归档	√	√
数据字典	√	√
内置任务调度	√	√
图形化管理工具	√	√

从表中可以看到，MySQL 在功能上已经与 SQL Server 没有太大差别了，而且比 SQL Server 有更快的性能、更灵活的扩展、更易用的管理工具以及更低的总体拥有成本。概括起来，MySQL 有如下特点：

（1）功能强大。MySQL 中提供了多种数据库存储引擎，各种引擎各有所长，适用于不同的应用场合，用户可以选择最合适的引擎以得到最高性能，可以处理每天访问量超过数亿的高强度的 Web 搜索站点。MySQL 5 支持事务、视图、存储过程、触发器等。

（2）支持跨平台。MySQL 支持至少 20 种以上的开发平台，包括 Linux、Windows、FreeBSD、IBMAIX、AIX 等。这使得在任何平台下编写的程序都可进行移植，而不需要对程序做任何修改。

（3）运行速度快。高速是 MySQL 的显著特性。尽管 MySQL 仍在开发中，但它已经提供一个丰富和极其有用的功能集。它的连接性、速度和安全性使 MySQL 非常适合访问在 Internet 上的数据库。在 MySQL 中，使用了极快的 B 树磁盘表和索引压缩，通过使用优化的单扫描多链接，能够实现极快的连接。SQL 函数使用高度优化的类库实现，运行速度极快。

（4）支持面向对象。PHP 支持混合编程方式。编程方式可分为纯粹面向对象、纯粹面向过程、面向对象与面向过程混合的 3 种方式。

（5）安全性高。灵活和安全的权限和密码系统，允许基本主机的验证。连接到服务器时，所有的密码传输均采用加密形式，保证了密码的安全。

（6）成本低。MySQL 数据库是一种完全免费的产品，用户可直接从网上下载。

（7）数据库存储容量大。MySQL 数据库的最大有效表尺寸通常由操作系统对文件大小限制决定，而不是由 MySQL 内部限制决定。InnoDB 存储引擎将 InnoDB 表保存在一个表空间内，该表空间可由数个文件创建，表空间的最大容量为 64TB，可以轻松处理拥有上千万条记录的大型数据库。

现在的 MySQL 已经不是大家眼中那个“小型”、“玩具”式的数据库系统了，已经可以和 SQL Server 等企业级数据库竞争了，而未来 MySQL 在新东家 Oracle 公司的带领下势必会让 MySQL 在企业级应用中更为稳定可靠，性能有更大的提升。

9.2　MySQL 的安装与初始化设置

9.2.1　获取 MySQL 安装包

虽然 MySQL 5.5.17 已经发布，但本书中以 MySQL 5.0.18 这个目前相对稳定的版本来

介绍。其他版本的 MySQL 安装方法也大致相同，即使读者的 MySQL 版本不是 5.0.18，也可以参照本节介绍的方法进行安装和初始化设置。

首先下载 MySQL 的安装包。可以直接从 MySQL 的官方网站来下载，网址为 http://www.mysql.com。也可以通过国内站点来下载，如 http://www.mysql.cn。由于本书中的例子都在 Windows 平台下进行开发和调试，因此要下载 Windows 平台下的 MySQL 安装包。笔者下载到的是一个名为 mysql-5.0.18-win32.zip 的压缩包。通过压缩包的名字就可以看出，这个 MySQL 版本为 5.0.18，并且适用于 Windows 32 位平台。

9.2.2　安装并配置 MySQL

MySQL 的安装过程如下。

解压安装包得到 setup.exe 的安装文件，启动安装程序后会出现软件安装欢迎界面，如图 9-1 所示。

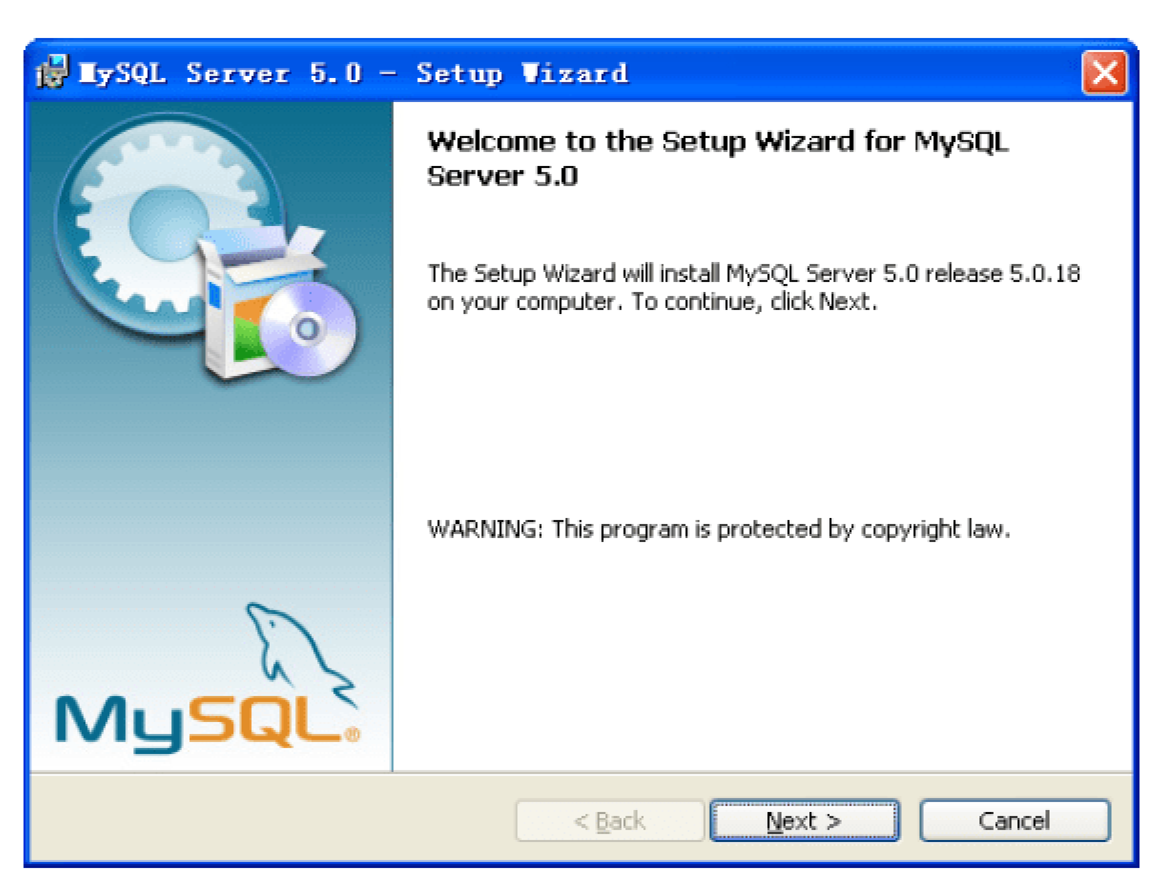


图 9-1　MySQL 数据库安装欢迎界面

单击 Next 按钮，出现安装类型选择窗口，如图 9-2 所示。默认选项为 Typical，即“典型”。这里建议选择 Custom 项，即“自定义”项。因为选择此项才能手工指定安装目录，否则将会安装到默认目录。

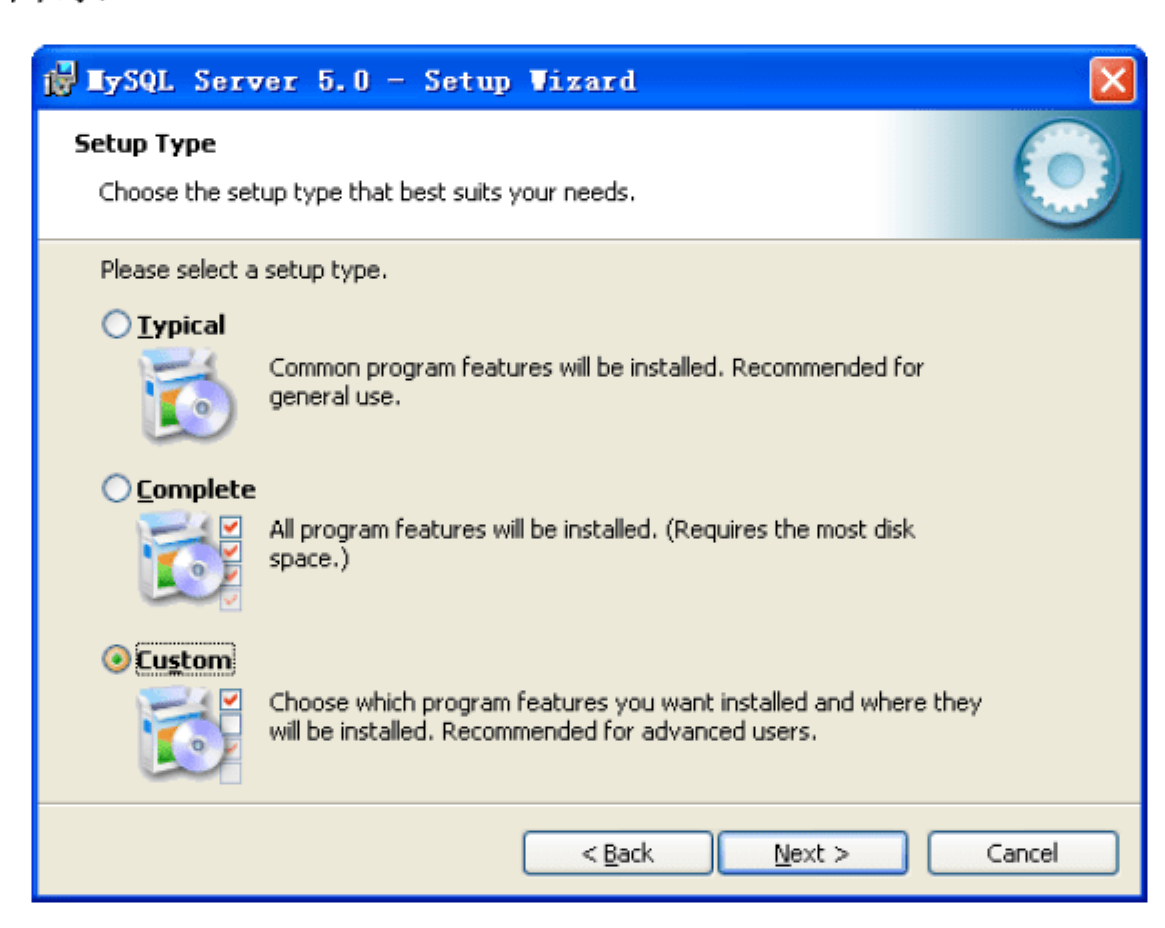


图 9-2　安装类型选择

单击 Next 按钮，出现“自定义安装”界面。在这里可以选择要安装的组件。因为安装全部组件也仅需要 23MB 的空间，因此建议不做改动，保持默认。窗口下方为安装位置，默认为“C:\Program Files\Mysql\Mysql Server5.0\”，可以看出这个目录较为复杂，不便使用，可以单击 Change 按钮，另外选择一个目录，如选择“C:\Mysql5\”作为安装目录，如图 9-3 所示。

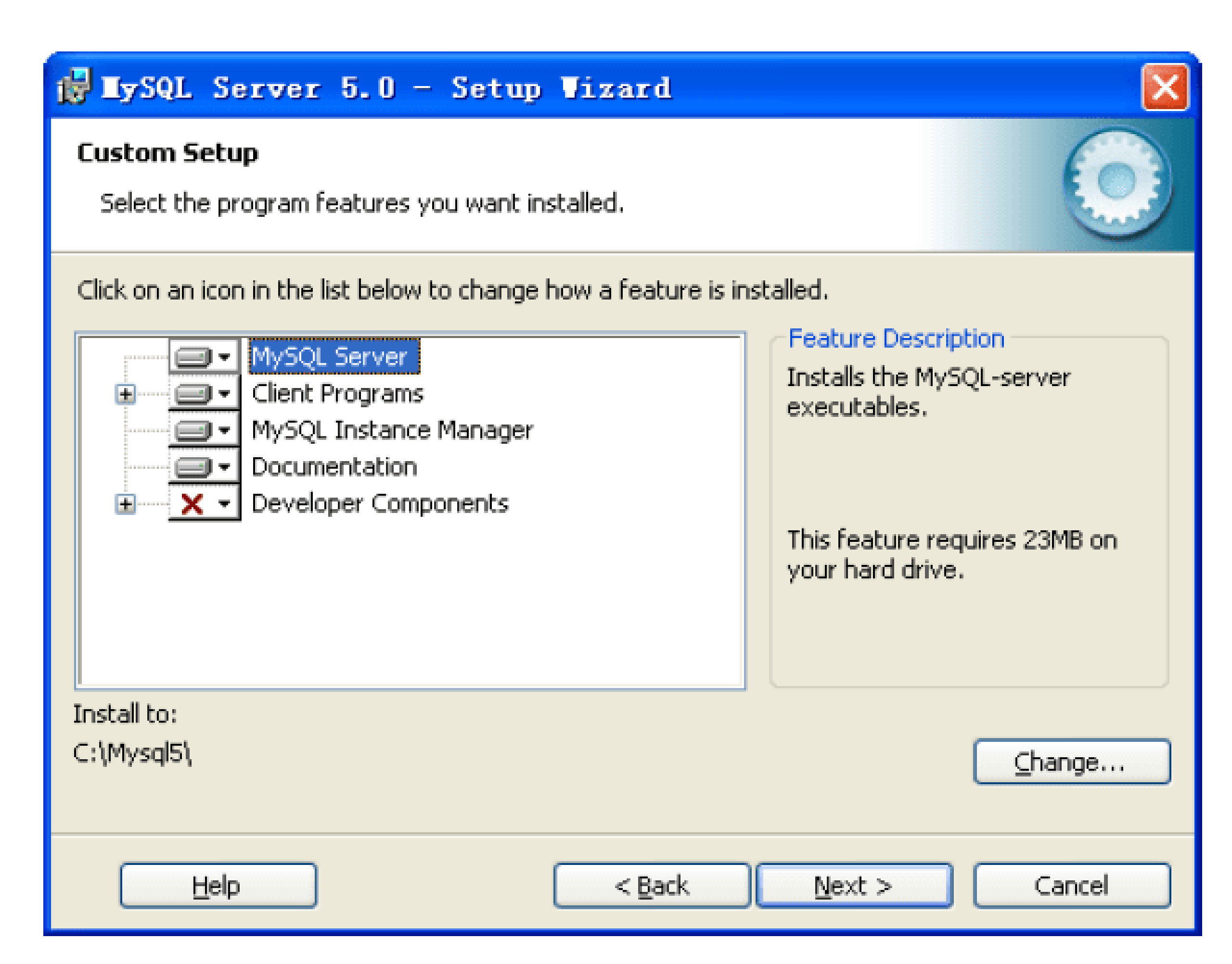


图 9-3　安装组件与目录选择

继续单击 Next 按钮，出现“准备就绪，确认安装”的界面，确认无误后单击 Install 按钮，此时出现安装进度条，如图 9-4 所示。安装进度完成后，会出现 MySQL 注册窗口，提示用户创建一个 MySQL 通行证或者登录 MySQL 通行证。可以选择 Skip Sign-Up 单选按钮直接跳过这一环节，如图 9-5 所示。

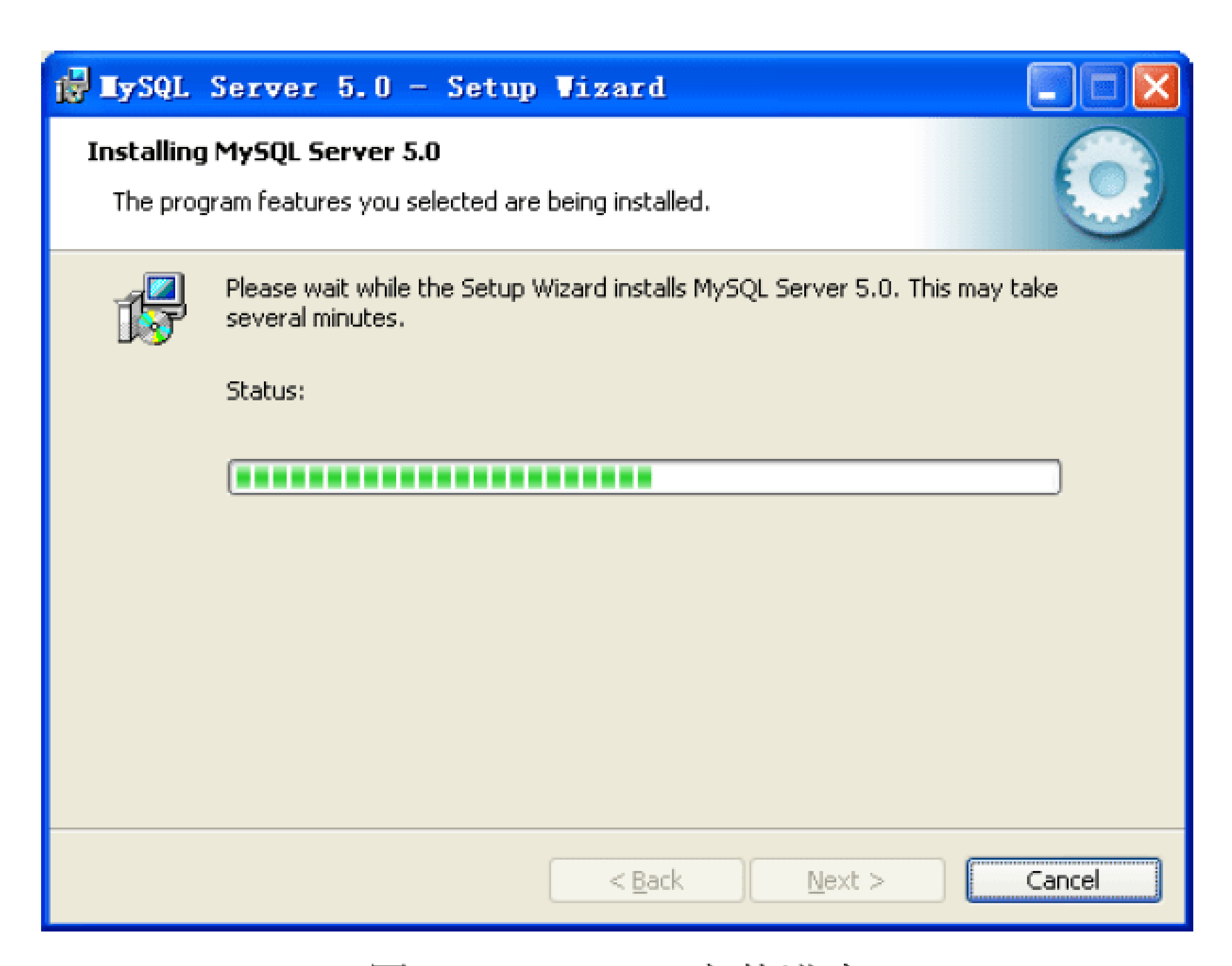


图 9-4　MySQL 安装进度

单击 Next 按钮，出现安装完毕窗口。这时有一个复选框，询问是否进行 MySQL 服务器配置。建议保持选中，并单击 Finish 按钮，这时可以启动 MySQL 服务器配置向导。

单击 Next 按钮，出现服务器配置类型窗口，保持默认的 Detailed Configuration，单击 Next 按钮，这时出现 3 个选项，可以选择 Developer Machine 或 Server Machine 项。这里选择 Developer Machine 项。单击 Next 按钮，出现选择数据库用途界面，可以选择 Multifunctional Database，即“多功能数据库”。单击 Next 按钮，出现 InnoDB 存放位置界面，保持默认直接单击 Next 按钮，出现并发连接数选择界面，如图 9-6 所示。这里第一个选项表示最大连接数为 20，第二个选项表示最大连接数为 500，第三个为自定义连接数。在学习阶段，建议选择第一个，20 个连接数足够使用。如果是真正配置 Web 服务器，可以根据需要选择更大的连接数。

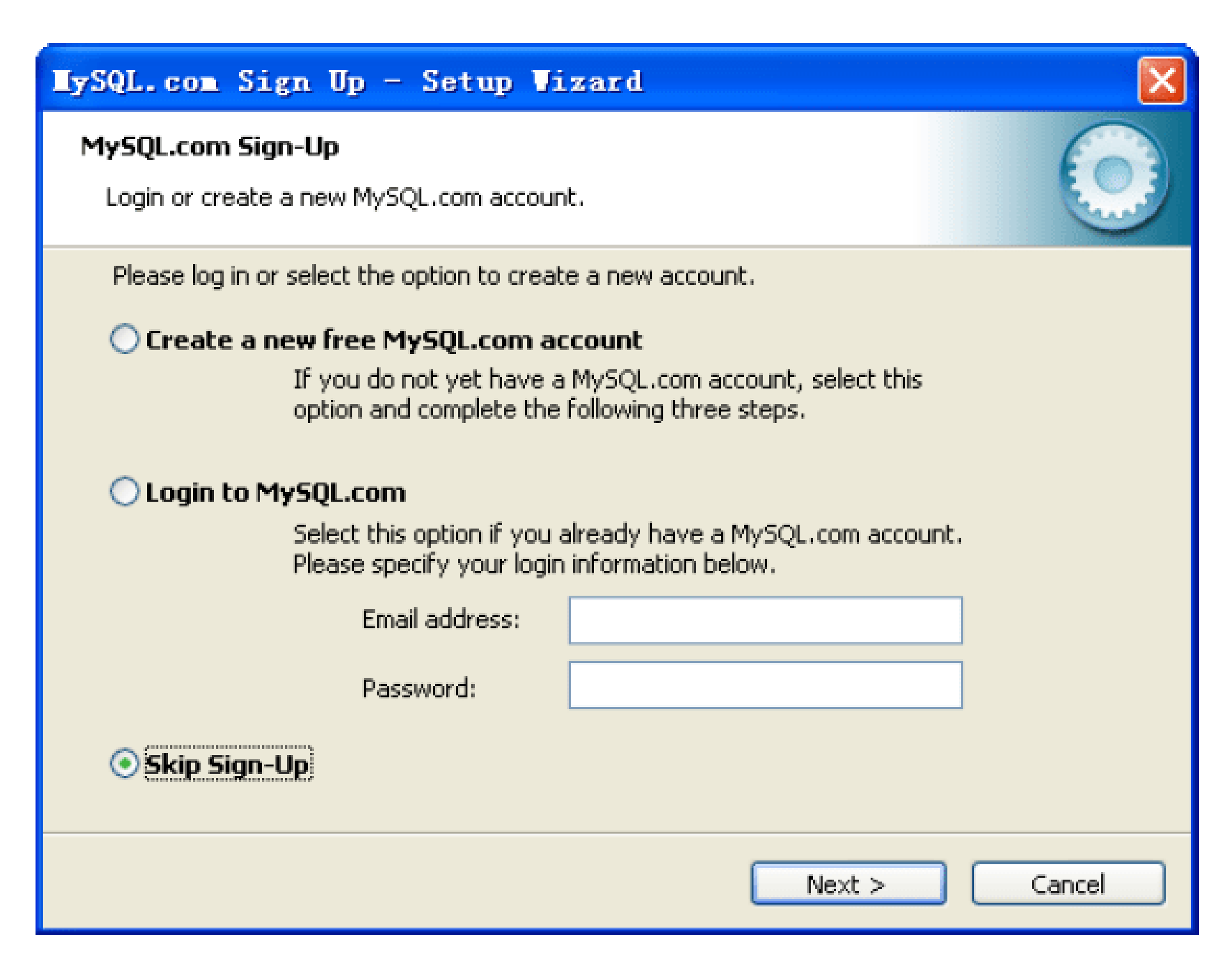


图 9-5　MySQL 联机注册

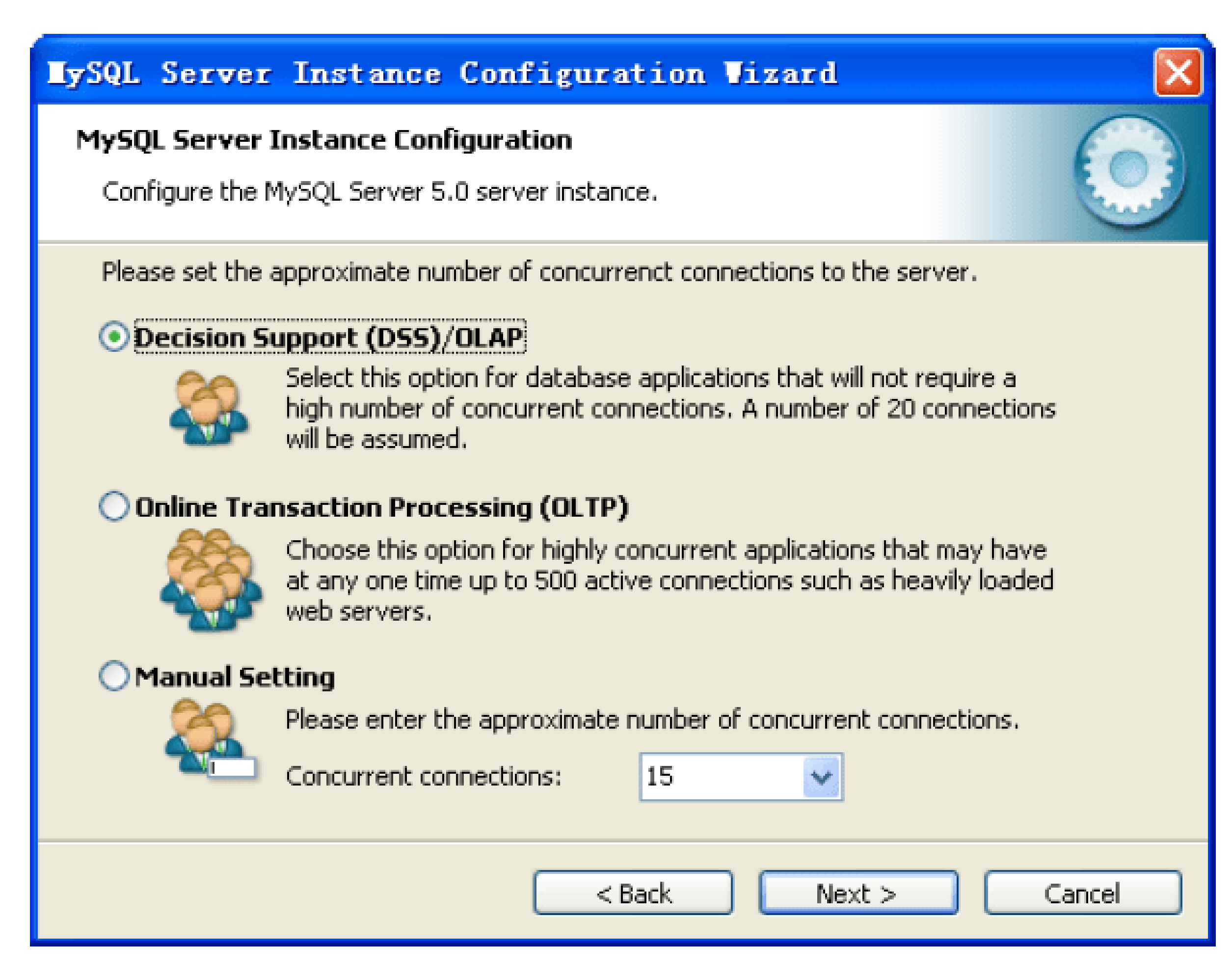


图 9-6　最大连接数选择

单击 Next 按钮，出现端口选择界面。这里就是 MySQL 数据库的服务端口，如果没有特殊需要，直接使用默认的 3306 端口即可。单击 Next 按钮，出现字符集设置界面，如图 9-7 所示。第一个选项为使用默认字符集，也就是把 Latin1 作为默认字符集，第二个选项为把 UFT-8 设置为默认字符集，第三个选项为自定义字符集。

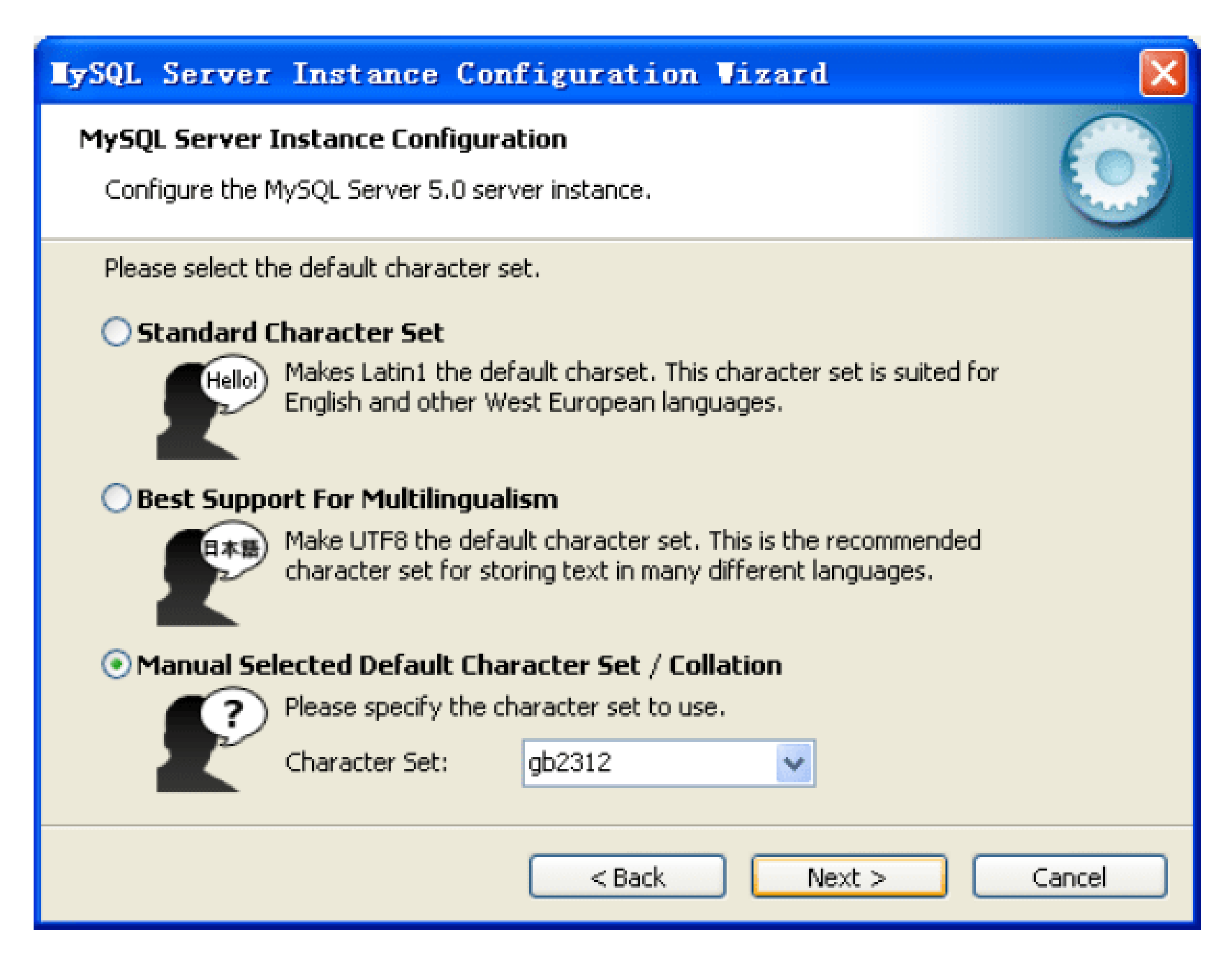


图 9-7 字符集设置

字符集的概念比较复杂，这里不再详述。事实上，采用什么样的字符集对 MySQL 影响不大，只有在不同 MySQL 之间导入导出数据时才考虑字符集是否一致的问题，否则容易导致乱码。这里不妨选择第 3 个选项，并把 GB2312 即简体中文设置为字符集（此处字符集的设置将对后期的 PHP+MySQL 调用产生影响，初学者可以保持默认，即采用 LATIN1）。

继续单击 Next 按钮，出现 Windows 选项界面，如图 9-8 所示。在这里可以选择是否将 MySQL 安装为 Windows 的服务。这里可以选择是，这样可以在计算机启动时自动启动 MySQL 数据库服务。另外下面的 Include Bin Directory in Windows PATH 也建议选中。本选项的意思是将 MySQL 的 Bin 目录加入到 Windows 的环境变量中。这样做可以让用户在命令行下直接运行 MySQL 命令，而无须先切换到 MySQL 的安装目录的 Bin 目录下（MySQL 的命令存放在安装目录下的 Bin 子目录下，如 C:\mysql5\bin）。



图 9-8 设置 Windows 选项

继续单击 Next 按钮，出现安全选项页面。在这个页面中可以设置 MySQL 数据库超级管理员的密码。MySQL 安装完毕之后默认生成一个用户名为 root 的超级管理员用户，密

码为空。这个用户拥有对数据库的完全控制权限。因此这个密码非常重要，一旦设置了就要务必牢记，一旦忘记很难找回。如果是在服务器上安装 MySQL，这个密码务必要设置，而且设置的越复杂越好。如果仅仅是在本地机器作为学习、测试之用，为了方便可以暂不设置密码。这里将密码设置为 1234。单击 Next 按钮，这时设置步骤完成，出现执行配置窗口，单击 Excute 按钮，开始执行配置，稍等片刻即可配置成功，如图 9-9 所示。

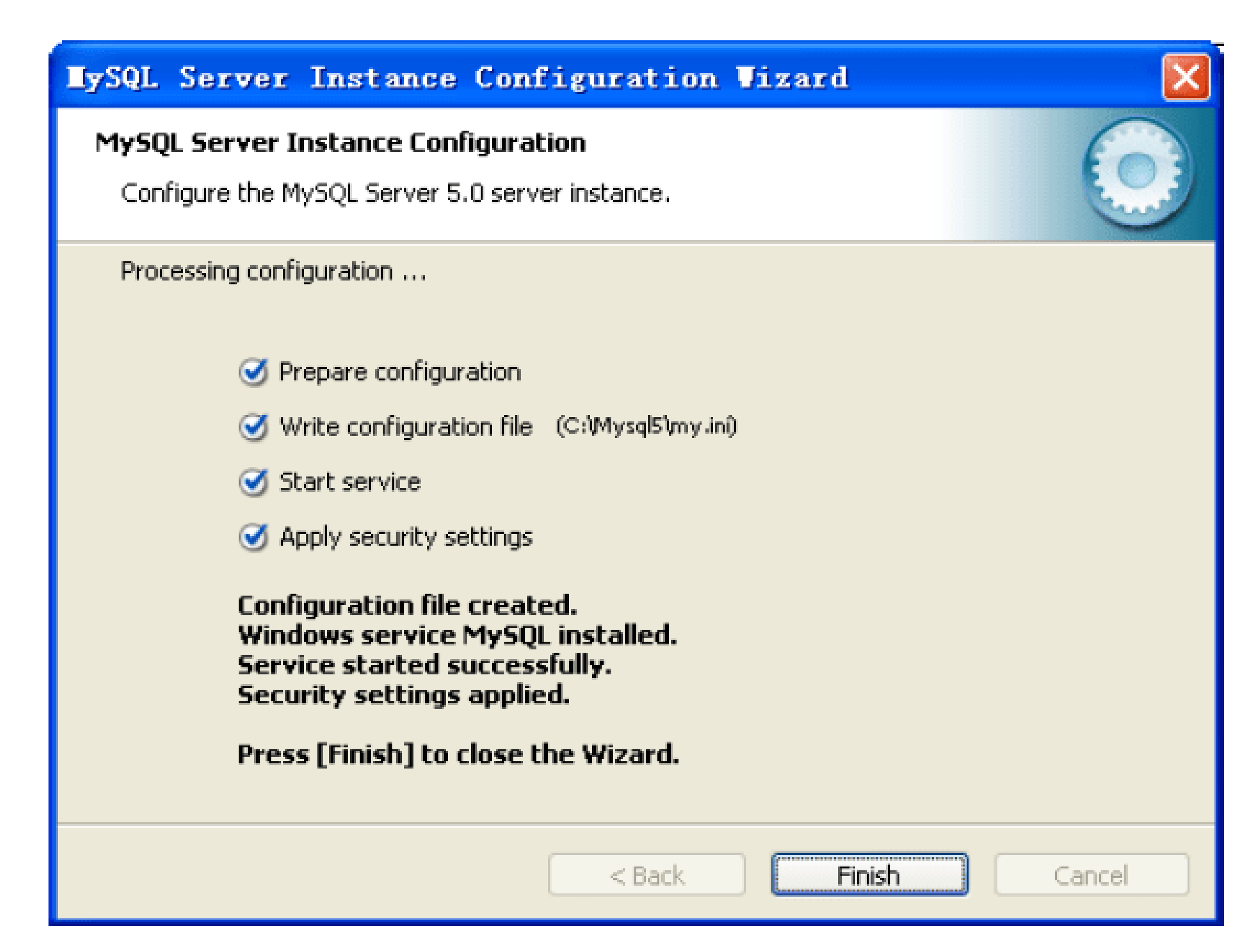


图 9-9　配置完成

这时，所有配置工作都已顺利结束，单击 Finish 结束配置程序。

9.2.3　进入 MySQL 控制台

如果安装配置成功，MySQL 服务应该已经被启动。可以通过以下方法进行简单的测试，来验证 MySQL 安装是否成功。

选择“开始”→“运行”命令，在“运行”对话框中输入执行 cmd 命令，打开命令提示符窗口，在命令提示符下输入“mysql□–u□root□-p”（其中□表示空格）并按 Enter 键，会出现 Enter password:，输入密码 1234 按 Enter 键，如果 MySQL 安装成功并已成功启动，会出现如图 9-10 所示的登录成功的欢迎信息。

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

C:\Documents and Settings\Administrator>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 5.0.18-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

图 9-10　MySQL 登录成功的欢迎信息

根据软件环境不同，命令提示符的显示信息可能也会有不同。但只要输入密码后能够看到“Welcome to the MySQL monitor.”之类的提示，就说明 MySQL 登录成功。如果 root

用户没有设置密码，可以在出现“Enter Password：”之后直接按 Enter 键，使用空密码进入。如果出现“MySQL 不是内部命令或外部命令，也不是可运行的文件或批处理文件”的提示，那说明在上面的配置步骤中，没有把 MySQL 的 Bin 目录加入到系统环境变量中。这时可以在命令提示符下切换到 MySQL 的安装目录的 Bin 目录，然后再输入“mysql-u root-p”命令登录 MySQL。

9.3 MySQL 中的数据类型

9.3.1 数据类型

在创建数据表的时候，需要指定要创建的每个列的数据类型。MySQL 支持大量的列类型，这些类型可以被分为 3 类：数值类型、时间日期类型和字符串类型。各大类中包含的具体类型及其取值范围如表 9-2 所示。

表 9-2 MySQL 支持的数据类型

大　类	数据类型	含义和取值范围或取值格式
数值型	TINYINT	很小的整数，有符号为－128～127，无符号为 0～255
	BOOL	同 TINYINT
	SMALLINT	小的整数，有符号为－32 768～32 767，无符号为 0～65 535
	MEDIUMINT	中等的整数，有符号为－8 388 608～8 388 607，无符号为 0～16 777 215
	INT	普通大小的整数，有符号为－2 147 483 648～2 147 483 647，无符号为 0～4 294 967 295
	INTEGER	同 INT
	BIGINT	大整数，有符号为－9 223 372 036 854 775 808～9 223 372 036 854 775 807，无符号为 0～18 446 744 073 709 551 615
日期时间型	DATETIME	时间和日期的组合，显示格式为 0000-00-00 00:00:00
	DATE	日期。显示格式为 0000-00-00
	TIMESTAMP	时间戳，00000000000000
	TIME	时间，格式为 00:00:00
	YEAR	年，默认格式为 0000
字符串型	CHAR	固定长度字符串，0～255（字节，字符型）
	VARCHAR	变长字符串，0～65 535（字节，字符型）
	BINARY	类似于 CHAR 类型，以二进制字节保存，0～255（字节，二进制型）
	VARBINARY	类似于 VARCHAR 类型，以二进制字节保存，0～65 535（字节，二进制型）

续表

大　类	数据类型	含义和取值范围或取值格式
字符串型	BLOB	无限大小（字节字符串）
	TEXT	无限大小（字符字符串）
	ENUM	枚举型，最多 65 535 个元素
	SET	集合型，最多 64 个成员

读者可能对表中的数据类型还很陌生。在后面的章节中将陆续介绍其中一些最为常用的类型。

9.3.2　字段属性

字段除了必须声明类型之外，还可以有各种属性。如有的字段值不能为空，有的字段可以设成“key（键）”，有的字段可以设成“Auto_increment 自增”，有的字段可以规定长度和设置默认值等。这就涉及到 MySQL 的字段属性。读者将在后面的学习中逐渐接触到不同的字段属性。

9.4　操作 MySQL 数据库

9.4.1　SQL 概述

结构化查询语言（Structured Query Language）是用来操作数据库的标准语言，最早是 IBM 的圣约瑟研究实验室为其关系数据库管理系统 SYSTEM R 开发的一种查询语言。SQL 结构简洁，功能强大，简单易学，所以自从 IBM 公司 1981 年推出以来，SQL 得到了广泛的应用，是关系数据库的通用语言，是数据库系统的工业标准。如今无论是 Oracle、Sybase、SQL Server 等大型的数据库管理系统，还是 Visual FoxPro、PowerBuilder 等桌面数据库开发系统，都支持 SQL 语言作为查询语言，MySQL 也不例外。SQL 可以完成的功能如下：

- 查询数据。
- 在表中插入、修改和删除记录。
- 建立、修改和删除数据对象。
- 控制对数据和数据对象的存取。
- 保证数据库一致性和完整性。

MySQL 不仅支持 SQL 语言，而且还对其进行了扩展，使得它能够支持更为强大的功能。下面就来介绍一些具体的 MySQL 支持的 SQL 语句，学习操作数据库中数据的方法。

9.4.2　操作数据库

1. 创建数据库

CREATE 语句可以用来创建新的数据库。根据本章前面讲过的方法，打开命令提示符界

面，输入用户名密码登录到 MySQL 控制台。登录到控制台后光标前面显示“mysql>”，在光标处可以直接输入 SQL 语句来创建一个名为 student 的数据库。输入以下命令并按 Enter 键：

```
mysql> create  database  student;
```

SQL 语句可以用大写，也可以用小写，还可以大小写混合。本语句执行后会输出：

```
Query OK, 1 row affected (0.08 sec)
```

这说明语句执行成功。一个名为 student 的数据库已经被创建成功。

2. 显示数据库

创建完数据库后，可以用 SHOW 语句来查看数据库是否已经被创建。

```
mysql> show databases;
```

输入命令后按 Enter 键，列出当前所有数据库。

```
mysql> show databases;
+--------------------+
|   Database         |
+--------------------+
| information_schema |
|    mysql           |
|   student          |
|    test            |
+--------------------+
4 rows in set (0.13 sec)
```

可以看到，student 数据库已经创建成功（information_schema、mysql、test 三个数据库均为 MySQL 安装时自动创建的原始数据库）。

3. 选择数据库

USE 语句用于选择一个数据库，使其成为当前数据库，语法如下：

```
use 数据库名;
```

4. 删除数据库

DROPDATABASE 语句用于删除一个数据库，语法如下：

```
drop database 数据库名;
```

如果是删除一个数据库，那么这个数据库下的所有表也将被删除。

9.4.3 操作数据表

1. 创建数据表

CREATE 语句不仅用于创建数据库，还用于在数据库中创建表。继续用 CREATE 语句在 student 数据库中创建一个表 info。这个表用来存储学生基本信息，一共有 3 个字段，分别是姓名（name）、性别（sex）、年龄（age）。这 3 个字段对应的数据类型分别为 CHAR、CHAR、TINYINT，长度分别限制在 20 字节、2 字节、2 字节以内。

在 student 数据库中创建表之前，需要首先打开这个数据库：

```
mysql> USE student;
```

此语句用 USE 命令选定一个要操作的数据库。执行后显示 Database changed，表示数据库已经打开。

然后输入以下语句并按 Enter 键：

```
mysql> create table info (name char(20), sex char(2), age tinyint(2));
```

注意：每条 SQL 语句输入完毕后最后要输入“;”，表示输入完成；否则不论输入多少个回车此语句都不会执行，直到遇到分号结尾（也有极个别语句可以不加分号）。

语句执行完毕后显示“Query OK, 0 rows affected (0.14 sec)”。表示语句执行成功。这时，表 info 已经创建成功。可以使用以下命令来查看 student 数据中现有的表。

```
mysql> SHOW TABLES;
```

执行后显示：

```
+--------------------+
| Tables_in_student |
+--------------------+
|   info            |
+--------------------+
1 row in set (0.00 sec)
```

这时，可以看到 info 表确实已经创建在 student 数据库中。

2. 往表中插入数据

INSERT 语句用来向表中插入新的数据记录。每次插入一条。如要向刚才创建的 info 表中插入一条各字段值分别为“张三”、“男”、“20”的记录，可以使用下面的语句：

```
mysql> insert into info values ("张三", "男", 20);
```

执行后显示“Query OK, 1 row affected (0.08 sec)”，表示语句执行成功。

值得注意的是，在插入数据时，字符串型值要用双引号或单引号引起来，数值型不用引号（加引号就错了）。而且，提供的数据也必须按照表的字段顺序排列，不能颠倒。

下面将介绍如何从表中查询数据。在查询之前，先执行几次 INSERT 语句向表中插入几条信息，这样可以更加形象地说明查询语句的作用。不妨再插入“李四”、“王五”、“赵六”3 条记录，这样表中共有 4 条记录。

3. 查询数据表

SELECT 语句用来查询表中的数据。SELECT 语句是 SQL 中最复杂的语句之一。因为用 SELECT 语句可以实现极为复杂的查询功能，如可以查询某个表中全部记录、部分满足条件的记录、全部字段、部分满足条件的字段等。还可以同时从多个表中查询满足条件的记录，以及对查询结果进行排序等。

这里仅介绍几种常用的 SELECT 语句，读者可以参考其他数据库专业书籍来更加深入地学习。

1）查询全部记录全部字段

查询一个表中全部记录，可以用如下语句：

```
mysql> select * from info;
```

这里“*”表示所有字段。info 为表名。程序执行后输出：

```
+------+------+------+
| name | sex  |  age |
+------+------+------+
|  张三 |  男  |   20 |
|  李四 |  男  |   18 |
|  王五 |  女  |   18 |
|  赵六 |  女  |   17 |
+------+------+------+
4 rows in set (0.02 sec)
```

可见刚才插入的 4 条数据全部被查询出来了。

2）查询全部记录的部分字段值

可以通过指定具体的字段和排序方式，来过滤不需要显示的字段。例如，要查询所有记录的姓名、年龄两个字段值，可以用如下语句：

```
mysql> select name,age from info;
```

执行后输出

```
+------+------+
| name | age  |
+------+------+
|  张三 |   20 |
|  李四 |   18 |
|  王五 |   18 |
|  赵六 |   17 |
+------+------+
4 rows in set (0.00 sec)
```

3）查询满足某个条件的记录

通过 SELECT 语句的 WHERE 子句，可以查询某些满足指定条件的记录，这在查询中极为常用。例如，要查询所有年龄小于 19 的记录，可以用如下语句：

```
mysql> select * from info where age<19;
```

执行后输出：

```
+------+------+------+
| name | sex  | age  |
+------+------+------+
|  李四 |  男  |   18 |
|  王五 |  女  |   18 |
|  赵六 |  女  |   17 |
+------+------+------+
3 rows in set (0.01 sec)
```

当查询条件有多个时，可以使用 AND 关键字。如现在查询所有年龄小于 19 并且性别为女的记录，可以使用下列语句：

```
mysql> select * from infor where age<19 and sex = "女";
```

该语句执行后，将只输出满足条件的“王五”、“赵六”的两条记录。

4）查询某些记录，并对结果进行排序

使用 SELECT 语句的 ORDER BY 子句可以对查询结果进行排序。如查询所有性别为“女”的记录，并且将结果按照年龄从小到大排序。

```
mysql> select * from info where sex= "女" order by age asc;
```

运行后输出结果如下：

```
+------+------+------+
| name | sex  |  age |
+------+------+------+
| 赵六 |  女  |  17  |
| 王五 |  女  |  18  |
+------+------+------+
2 rows in set (0.01 sec)
```

如果要将从小到大改为从大到小，则将命令中的 asc 改为 desc 即可。

4. 修改数据表

（1）ALTER 语句用来修改一个表的定义，也就是说修改表自身，如修改表的名字，修改表中某个字段的名字、属性、类型等（也可以用于修改数据库的部分属性）。看下面的例子：

```
mysql> alter table info change name xingming char (20);
```

本语句将表 info 的 name 字段名修改为 xingming，类型和长度不变。又如：

```
mysql> alter table info add addr char (50);
```

本语句在 info 表中又增加了一个名为 addr，类型 char，长度为 50 的新字段。

```
mysql> alter table info drop addr;
```

本语句删除了表 info 中的 addr 字段。

（2）UPDATE 语句可以对表中现有的记录进行修改。

① 修改全部记录的某个字段的值。

例如，要将 info 表中全部记录的年龄都修改成 25，可以使用下面的语句：

```
mysql> update info set age=25;
```

这时，如果用 SELECT 语句查询此表，会看到所有记录的 age 字段都变成了 25。读者可以执行 SELECT * FROM info;语句来查看表中的数据，SELECT 语句的详细用法将在 9.4.4 节介绍。

此外，还可以一次修改多个字段的值。如除了要将所有记录的 age 字段修改成 25，还要将所有 sex 记录修改为“女”，可以用如下语句：

```
mysql> update info set age = 25, sex ="女";
```

也就是说，多个字段之间用逗号隔开，可以一次修改多个字段的值。

② 修改满足某条件的记录。

通过 WHERE 子句指定的条件，可以修改满足指定条件的记录的值。如要将姓名为“张三”的记录的年龄修改成 23，可以用如下语句：

```
mysql> update info set age = 23 where name = "张三";
```

执行之后再用 SELECT 语句查询此表，会发现“张三”的年龄为 23，其他记录的年龄的均为 25。

同样可以用逗号隔开的方法，修改满足指定条件的记录的多个字段。

（3）删除表中的记录。

DELETE 语句用来删除表中的记录。可以一次删除全部记录，也可以删除满足指定条件的记录。

① 删除表中的全部记录。

如要删除表 info 中全部记录，可以用以下语句：

```
mysql>delete from info;
```

该语句执行后表 info 中的全部记录都会被删除。可以看出，该命令是比较危险的，不小心很容易造成误删。因此使用此命令时应尽量注意。

② 删除满足条件的记录。

如果要删除表 info 中性别为“女”的记录，可以用如下命令：

```
mysql> delete form info where sex = "女";
```

读者可以自行尝试变换条件，来观察语句运行效果。

5. 删除数据表

DROP 语句用来删除一个数据表。例如：

```
mysql> drop table tbl1;
```

删除名为 tbl1 的表（删除前需要先打开数据库）。

9.5 MySQL 数据库的管理

9.5.1 MySQL 的用户管理

MySQL 的用户管理，指的是哪个用户可以连接服务器，从哪里连接，连接后能做什么。MySQL 用户包括普通用户和 root 用户，这两种用户的权限是不一样的。root 用户是超级管理员，拥有所有的权限。root 用户的权限包括创建用户、删除用户、修改普通用户的密码等管理权限。而普通用户只拥有创建该用户时赋予它的权限。用户管理包括管理用户的账户、权限等。安装 MySQL 时会自动安装一个名为 MySQL 的数据库，用户账号和密码以及权限等信息，都存储在这个名为 MySQL 的数据库的 user 表中（MySQL 安装完成后自动创建，可以在控制台下查看）。MySQL 数据库下面存储的都是权限表。用户登录以后，MySQL 数据库系统会根据这些权限表的内容为每个用户赋予相应的权限。这些权限表中最重要的是 user 表、db 表和 host 表。除此之外，还有 tables_priv 表、columns_priv 表、proc_priv 表等。

分别执行以下两个命令：

```
mysql> use mysql
mysql> select * from user;
```

这时，可以看到类似于下面样式的返回结果（以下结果进行过简化）。

```
+----------+-------+--------------+---------+---------+---------
| Host     | User  | Password  ……
| %        | root  | 1c8bc9fa64c40b82  ……
+----------+-------+--------------+---------+---------+---------
1 rows in set (0.00 sec)
```

可以看到，查询出 user 表中的记录，每条记录就是一个用户账号信息。由于 user 表有数十个字段，因此读者看到的查询结果可能显示的比较零乱，这是由于屏幕尺寸有限，无法在一行内显示出所有字段，自动换行后导致的。

新安装的 MySQL，一般 user 表中有两个用户，分别是 root 和匿名用户。匿名用户即不需要用户名和密码即可进入系统的用户。

在 user 表中，前 3 个字段 Host、User、Password 分别表示登录主机、用户名和密码。登录主机表示此用户允许登录的主机地址，即 IP 地址。“%”表示任意主机。如果本用户只能从本地登录，不允许远程登录，可以用 localhost 或本机 IP 地址。用户密码用加密方式存储，因此看到的密码是一串无规则的字符串。从第 4 个字段以后的字段，表示权限状态，即该用户是否有某权限。这些权限包括查询权限、修改权限、删除权限等。

user 表是 MySQL 中最重要的一个权限表。读者可以使用 DESC 语句查看 user 表的基本结构。user 表有 39 个字段。这些字段大致可以分为 4 类，分别是用户列、权限列、安全列和资源控制列。

db 表和 host 表也是 MySQL 数据库中非常重要的权限表。db 表中存储了某个用户对一个数据库的权限。db 表比较常用，而 host 表很少会用到。可以使用 DESC 语句查看这两个表的基本结构。这两个表的表结构差不多。db 表和 host 表的字段大致可以分为两类，分别是用户列和权限列。

tables_priv 表可以对单个表进行权限设置。columns_priv 表可以对单个数据列进行权限设置。可以使用 DESC 语句来查看这两个表的基本结构。tables_priv 表包含 8 个字段，分别是 Host、Db、User、Table_name、Table_priv、Column_priv、Timestamp 和 Grantor。前 4 个字段分别表示主机名、数据库名、用户名和表名。Table_priv 表示对表进行操作的权限。这些权限包括 Select、Insert、Update、Delete、Create、Drop、Grant、References、Index 和 Alter。Column_priv 表示对表中的数据列进行操作的权限。这些权限包括 Select、Insert、Update 和 References。Timestamp 表示修改权限的时间。Grantor 表示权限是谁设置的。

procs_priv 表可以存储过程和存储函数进行权限设置。可以使用 DESC 语句查看 procs_priv 表的基本结构。procs_priv 表包含 8 个字段，分别是 Host、Db、User、Routine_name、Routine_type、Proc_priv、Timestamp 和 Grantor。前 3 个字段分别表示主机名、数据库名和用户名。Routine_name 字段表示存储过程或函数的名称。Routine_type 字段表示类型。该字段有两个取值，分别是 FUNCTION 和 PROCEDURE。FUNCTION 表示这是一个存储函数。PROCEDURE 表示这是一个存储过程。Proc_priv 字段表示拥有的权限。权限分为 3 类，分别是 Execute，Alter Routine 和 Grant。Timestamp 字段存储更新的时间。Grantor 字

段存储权限是谁设置的。

在 MySQL 数据库中，可以使用 CREATE USER 语句来创建新的用户，也可以直接在 mysql.user 表中添加用户。还可以使用 GRANT 语句来新建用户。本小节将为读者介绍这 3 种方法。

1. 用 CREATE USER 语句新建普通用户

语法如下：

```
create user 'username'@'host' identified by 'password';
```

说明： username 为将创建的用户名，host 指定该用户在哪个主机上可以登录，如果是本地用户可用 localhost，如果想让该用户可以从任意远程主机登录，可以使用通配符%。password 为该用户的登录密码，密码可以为空，如果为空则该用户可以不需要密码登录服务器。

例如：

```
create user 'dog'@'localhost' identified by '123456';
create user 'pig'@'192.168.1.101_' idendified by '123456';
create user 'pig'@'%' identified by '123456';
create user 'pig'@'%' identified by '';
create user 'pig'@'%';
```

2. 用 INSERT 语句新建普通用户

语法如下：

```
mysql>use mysql;
mysql>insert into mysql.user(主机名称,用户名称,用户密码,权限 1,权限 2, …)
values(host,user,password,select_priv,…);
mysql>flush privileges;
```

例如：

```
mysql>use mysql;
mysql> insert into user (host,user,password) values ('localhost','jsjdpt',password
('112233'));
mysql>flush privileges;   //刷新系统权限表
```

这样就创建了一个名为 jsjdpt，密码为 112233 的用户。

3. 用 GRANT 语句新建普通用户

最好用的方法是使用 GRANT 语句，因为这样更精确，错误少。从 MySQL 3.22.11 起提供了 GRANT；它的主要用途是来给账户授权的，但也可用来建立新账户并同时授权。语法如下：

```
grant priv_type [(column_list)] [, priv_type [(column_list)] ...]
on {tbl_name | * | *.* | db_name.*}
to user_name [identified by 'password']
   [, user_name [identified by 'password'] ...]
[with grant option]
```

这是完整的 GRANT 语句语法结构，看起来比较复杂。使用本命令可以一次创建多个 MySQL 账号。在实际应用中一般一次只创建一个用户，这样语法结构就可以简化为：

```
grant  priv_type [(column_list)]
on  {tbl_name | * | *.* | db_name.*}
to  user_name  [identified by 'password']
```

而到了具体的语句中，还可以继续简化。例如：

```
mysql> grant all on db1.* to "nie" identified by "123456";
```

此语句执行之后创建用户 nie，密码 123456，该用户对数据库 db1 拥有全部权限。

下面对 grant 语句的语法结构进行简要分析。

（1）grant：关键字，表示授权语句开始。

（2）priv_type：权限类型。可以是 select/delete/update/create/drop/alter 等任意一种。如果是全部权限，可以用 all privileges，并且可以简写为 all。

（3）on { tbl_name | * | *.* | db_name.*}：声明此用户可以操作哪些数据库以及哪些表。声明可以使用以下 4 种方法之一：

- tbl_name：直接指定表名，如 info。
- *：任意表。
- *.*：任意数据库的任意表。
- db_name.*：指定数据库的所有表，如 db1。

（4）to user_name：指定用户名。即要创建的账号的用户名，如上例中的 nie。

（5）identified by：此项目为可选。指定账号所对应的密码。应用引号引起来。密码提交后会自动被加密。

例如：

```
mysql>grant select,insert,update,delete,create,drop on student.course to
zhangsan@219.218.22.187 identified by '1234';
```

给来自 219.218.22.187 的用户 zhangsan 分配可对数据库 student 的 course 表进行 select,insert,update,delete,create,drop 等操作的权限，并设定密码为 1234。

```
mysql>grant all privileges on student.* to zhangsan@219.218.22.187 identified
by '1234';
```

给来自 219.218.22.187 的用户 zhangsan 分配可对数据库 student 所有表进行所有操作的权限，并设定密码为 1234。

9.5.2　删除用户和取消权限

数据库管理员不仅能创建用户并赋予权限，当然也能取消用户授权和删除用户。REVOKE 语句用于取消某个用户的权限，语法如下：

```
revoke privileges(columns) on what from user;
```

其中，privileges 是要取消的权限，user 是要被取消权限的用户。例如：

```
mysql>revoke all on *.* from zhangsan@localhost;
query ok, 0 rows affected(0.00sec)
```

revoke 语句只能取消用户权限，不能删除用户。即使取消了某个用户的所有权限，该用户依然可以连接到服务器。要删除用户，需使用 delete 语句，将该用户的记录从 MySQL

数据库的 user 表中删除。语法如下：

```
delete from user where user="user_name" and host="host_name";
```

例如：

```
mysql>use mysql
database changed
mysql>delete from user where user="zhangsan" and host="localhost";
mysql>flush privileges;
query ok, 1 row affected(0.02sec)
```

删除了本地机器上的用户 zhangsan。

9.5.3 MySQL 的密码管理

1. root 用户修改自己的密码

root 用户拥有很高的权限，因此必须保证 root 用户的密码的安全。root 用户可以通过多种方式来修改密码。本小节将介绍几种 root 用户修改自己的密码的方法。

（1）使用 mysqladmin 命令来修改 root 用户的密码，步骤如下：

选择“开始”→“运行”命令，在弹出的对话框中输入 cmd 命令，单击“确定”按钮，进入命令行模式，输入：

```
mysqladmin -u 用户名 -p 原密码 password 新密码
```

按 Enter 键。注意，-u 和用户名，-p 和原密码之间的空格可以省略，但 password 和新密码之间的空格不能省略。

（2）修改 MySQL 数据库下的 user 表。

语法：

```
mysql>update mysql.user set password=password('新密码') where user="zhangsan"
and host="localhost";
```

例如：

```
mysql>update user set password=password('54321') where user='root';
mysql>flush privileges; //刷新系统权限表
```

（3）使用 SET 语句修改 root 用户的密码。

```
mysql>set password for 'username'@'host' = password('新密码');
```

如果是当前登录用户用

```
set password = password("新密码");
```

例如：

```
mysql>set password for root@localhost=password('');
mysql>set password for name=password('new password');
mysql>set password for 'pig'@'%' = password("123456");
```

2. root 用户修改普通用户密码

root 用户不仅可以修改自己的密码，还可以修改普通用户的密码。root 用户登录 MySQL

服务器后，可以通过 SET 语句、修改 user 表和 GRANT 语句修改普通用户的密码。本小节将向读者介绍 root 用户修改普通用户密码的方法。

（1）使用 SET 语句来修改普通用户的密码。

```
mysql>set password for 'user'@'localhost'=password("new_password");
```

（2）修改 mysql 数据库下的 user 表。

```
mysql>update mysql.user set password=password("new_password") where user=
''and host='';
mysql>flush privileges;
```

（3）用 GRANT 语句来修改普通用户的密码。

```
mysql>grant priv_type on database.table to user identified by [password]'
password[,user[identified by [password]'password']]...
```

3. 普通用户修改密码

普通用户也可以修改自己的密码。这样，普通用户就不需要每次需要修改密码时都通知管理员。普通用户登录到 MySQL 服务器后，可以通过 SET 语句来设置自己的密码。SET 语句的基本形式如下：

```
mysql>set password=password('new_password');
```

4. root 用户密码丢失的解决办法

如果 root 用户密码丢失，会给用户造成很大的麻烦。但是，可以通过某种特殊方法登录到 root 用户下。然后在 root 用户下设置新的密码。下面是解决 root 用户密码丢失的方法，执行步骤如下：

（1）KILL 系统中的 MySQL 进程：

```
mysql>killall -term mysqld
```

（2）以不检查权限的方式启动 MySQL，启动指令如下：

```
mysql>safe_mysqld --skip-grant-tables &
```

（3）然后用空密码方式，以 root 用户方式登录 MySQL：

```
mysql>mysql -u root
```

（4）修改 MySQLroot 用户的密码：

```
mysql> update mysql.user set password=password('新密码')
       where user='root';
mysql> flush privileges;
mysql> quit
```

重新启动 MySQL，就可以使用新密码登录了。

9.6 MySQL 的数据管理

数据库中的数据非常重要，以至于本节不得不将有关数据保护的方法从数据库管理中摘出来，用专门的章节来讲解如何保护数据库中的数据。

9.6.1 MySQL 的备份

备份是最简单的保护数据的方法，通过对数据库的备份，可以有效地保护数据，防止系统在遭到破坏时数据丢失，是数据库管理员必须掌握的技术。数据越是重要，数据的变化越频繁，备份越是需要经常进行。当数据库受到损坏时，可以用备份恢复原来的数据。

1. 使用 mysqldump 命令备份

mysqldump 是逻辑备份。mysqldump 命令能将整个数据库以 sql 语句的方式导出到一个 sql 文本文件中。mysqldump 不仅能备份表结构和数据，还可以同时支持 myisam 和 innodb 引擎数据库。mysqldump 可以备份单个表、单个库或所有库，还可以只导出表结构。最常见的 mysqldump 命令是制作整个数据库的一个备份：

```
mysqldump -opt database>backup.sql
```

如备份 student 数据库，操作如下：

选择“开始”→“运行”命令，在打开的对话框中输入执行 cmd 命令，单击“确定”按钮，打开命令提示符窗口，在命令提示符下直接输入

```
mysqldump -uroot -p1234 -opt student>d:\ab.sql
```

按 Enter 键即可。其中，-u 和-p 表示连接的用户名和密码，数据文件被备份到 d 盘的 ab.sql 中。

2. 直接复制数据库文件

先关闭 MySQL 服务器，不允许其他用户进行任何更新操作，然后直接把*.frm 表的描述文件、*.MYD 表的数据文件、*.MYI 表的索引文件复制到备份的目录下。

9.6.2 MySQL 的恢复

1. 用 mysqldump 备份的数据库的恢复

选择“开始”→“运行”命令，在打开的对话框中输入执行 cmd 命令，单击“确定”按钮，打开命令提示符窗口，在命令提示符下直接输入

```
mysql -uroot -p1234 student
```

进入到数据库中，然后在其后面输入备份的数据库所在的路径即可。其中，-u 后的 root 代表用户名，-p 后的 1234 代表密码，student 代表数据库名。例如：

```
mysql -uroot -p1234 student<d:\jsj\course.sql
```

将 D:\jsj\路径下的 course.sql 中的数据恢复到 student 数据库中。如果要恢复的是整个数据库，则直接输入数据库的名称；如果要恢复的是数据库中的某个表，则要输入数据库的库名和表名。

2. 直接复制数据库文件

先关闭 MySQL 服务器，不允许其他用户进行任何更新操作，然后直接把复制出来的*.frm 表的描述文件、*.MYD 表的数据文件、*.MYI 表的索引文件再复制到原来的目录下，重启 MySQL。

9.7　MySQL 可视化管理工具——phpMyAdmin

9.7.1　phpMyAdmin 的安装

在命令提示符下编写语句进行数据库管理，虽然能够比较灵活地对数据进行操控，但是却具有难度高、效率较低、容易出错等弊端。很多对命令提示符不熟悉的读者更是难以接受。实际上还有更加高效和便捷的方法来管理数据库，如 phpMyAdmin 就是一个专门用来管理 MySQL 数据库的图形界面工具，也是目前应用最为广泛的 MySQL 数据库管理工具。

phpMyAdmin 不是一般的桌面应用软件，是完全用 PHP 开发的一套软件，可以安装在安装了 PHP 和 MySQL 的计算机上，通过浏览器管理 MySQL 数据库。该软件功能强大，界面友好，深受 PHP+MySQL 开发者的青睐。phpMyAdmin 的压缩包可以从网络上下载，通过搜索引擎很容易就能找到其下载地址，在 http://www.mysql.cn 也可以下载到。该软件有诸多版本，在此以 2.8.0 版本为例说明其使用方法。

将下载到的压缩包解压缩，解压之后能得到一个包含了大量 PHP 程序的文件夹，此文件夹一般名为 phpMyAdmin-xx.xx.xx。为了将来使用方便，可以将此文件夹重命名，不妨命名为 phpmyadmin。

将 phpmyadmin 文件夹复制到 IIS 或 Apache 主目录下。这样就可以通过 http://localhost/phpmyadmin/运行此程序。

打开 phpmyadmin/scripts/文件夹，找到 config.default.php，将其修改为 config.inc.php（在不同版本下此文件的存放位置可能略有不同，如有的版本此文件直接存放在根目录下，如找不到可使用搜索功能）将此文件复制到 phpmyadmin/目录下，然后打开此文件，找到“$cfg['Servers'][$i]['password']=”项，将其值设置为 MySQL 的超级用户密码，如图 9-11 所示。

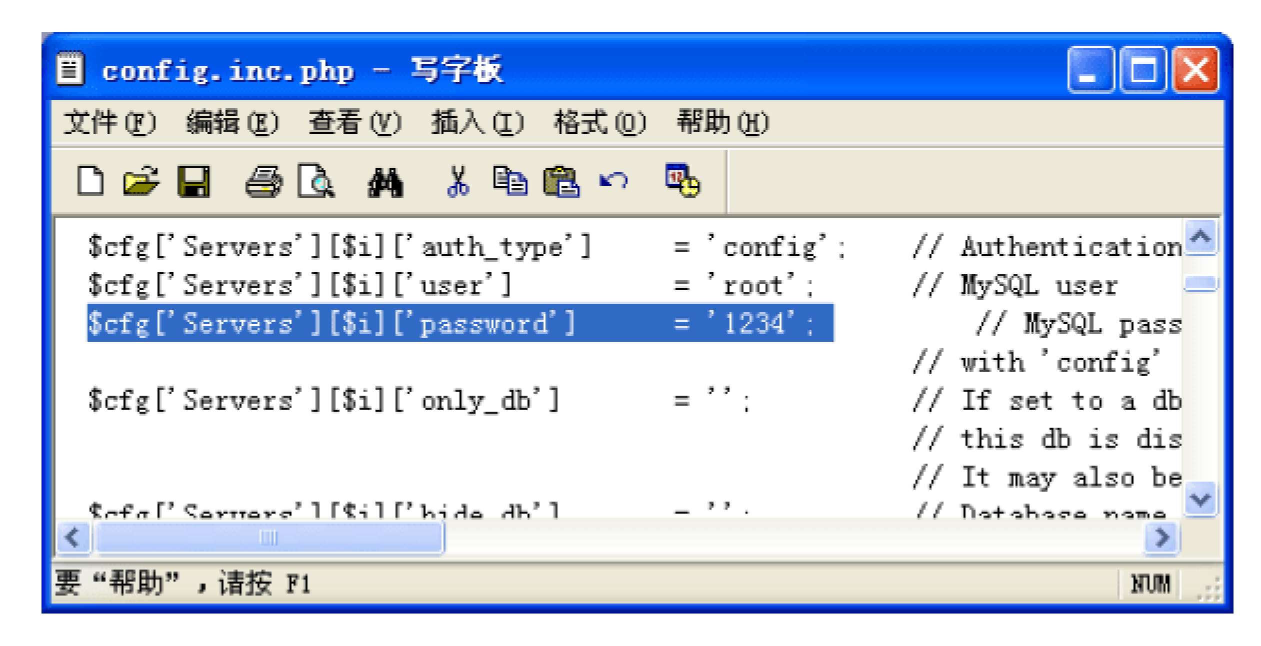


图 9-11　修改 phpmyadmin 配置文件

如果超级用户用户名不是 root，还需要将此文件中的“$cfg['Servers'][$i]['user']”修改为超级用户的用户名。

修改完毕之后，保存文件。这时，在浏览器中输入 http://localhost/phpmyadmin/index.php，就打开了 phpmyadmin 的管理界面，如图 9-12 所示。

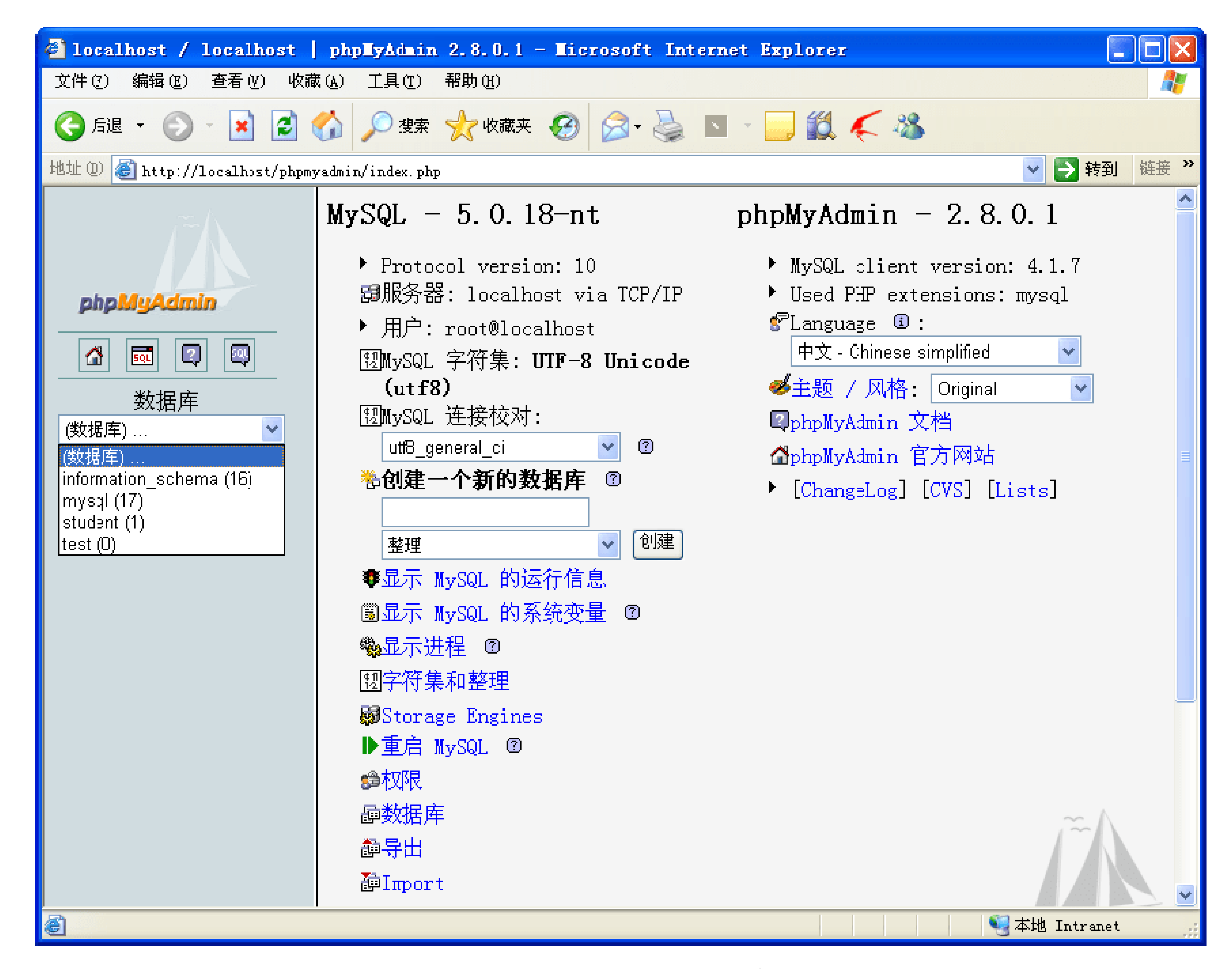


图 9-12　phpMyAdmin 运行界面

9.7.2　phpMyAdmin 的使用

1. 创建新数据库

要创建一个新数据库，在管理界面右侧的“创建一个新数据库”下的表单中输入数据库的名字，单击“创建”按钮，即可创建一个新数据库。

2. 选择数据库

在左侧的下拉菜单中，单击选择一个数据库，那么该数据库就被选中了，下面将列出该数据库中的所有表，右侧将列出数据库中表的信息，如图 9-13 所示。

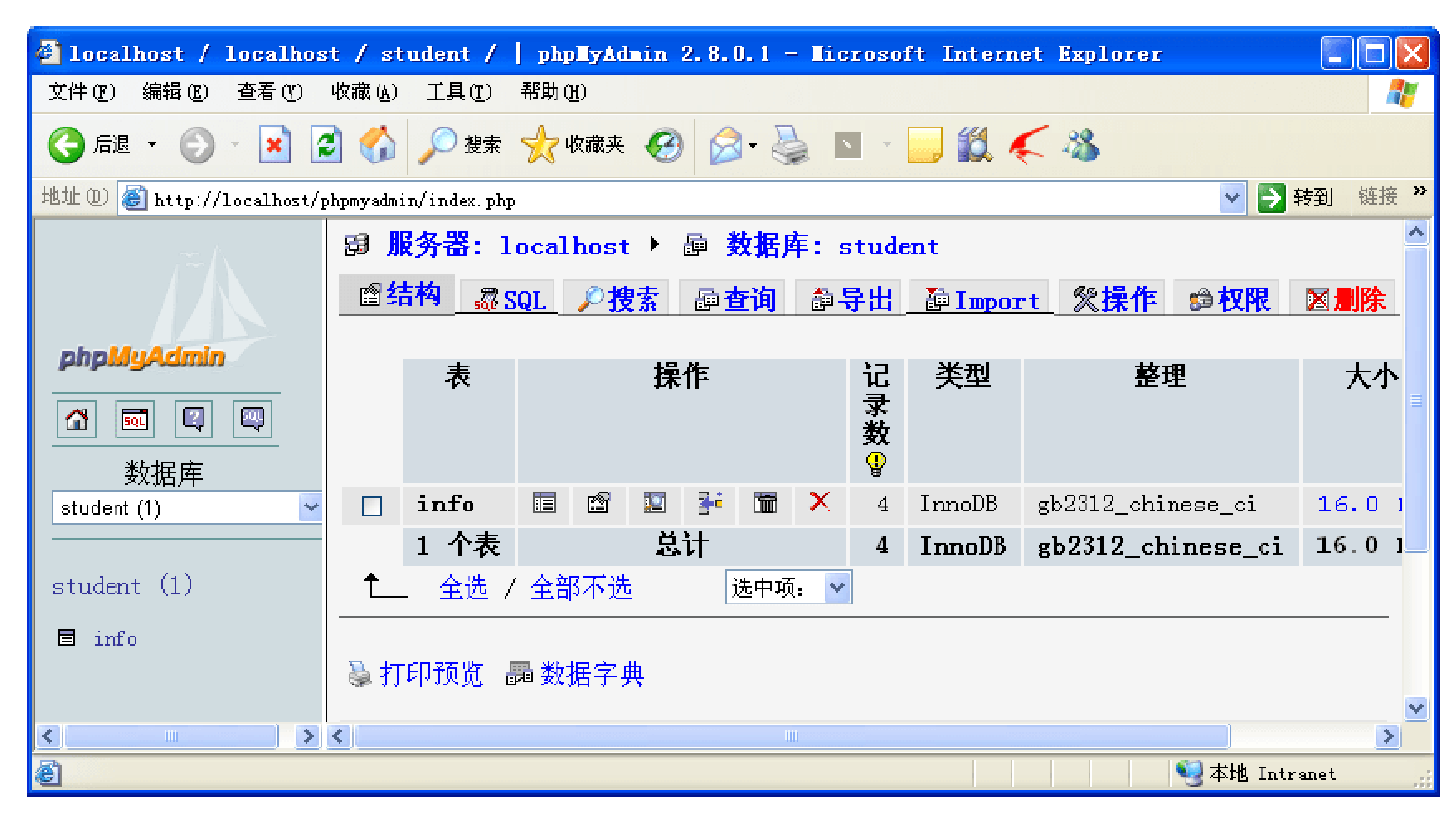


图 9-13　选择数据库

3. 选择并浏览表信息

单击左侧的表名，右侧将显示此表的详细字段信息，如图 9-14 所示。

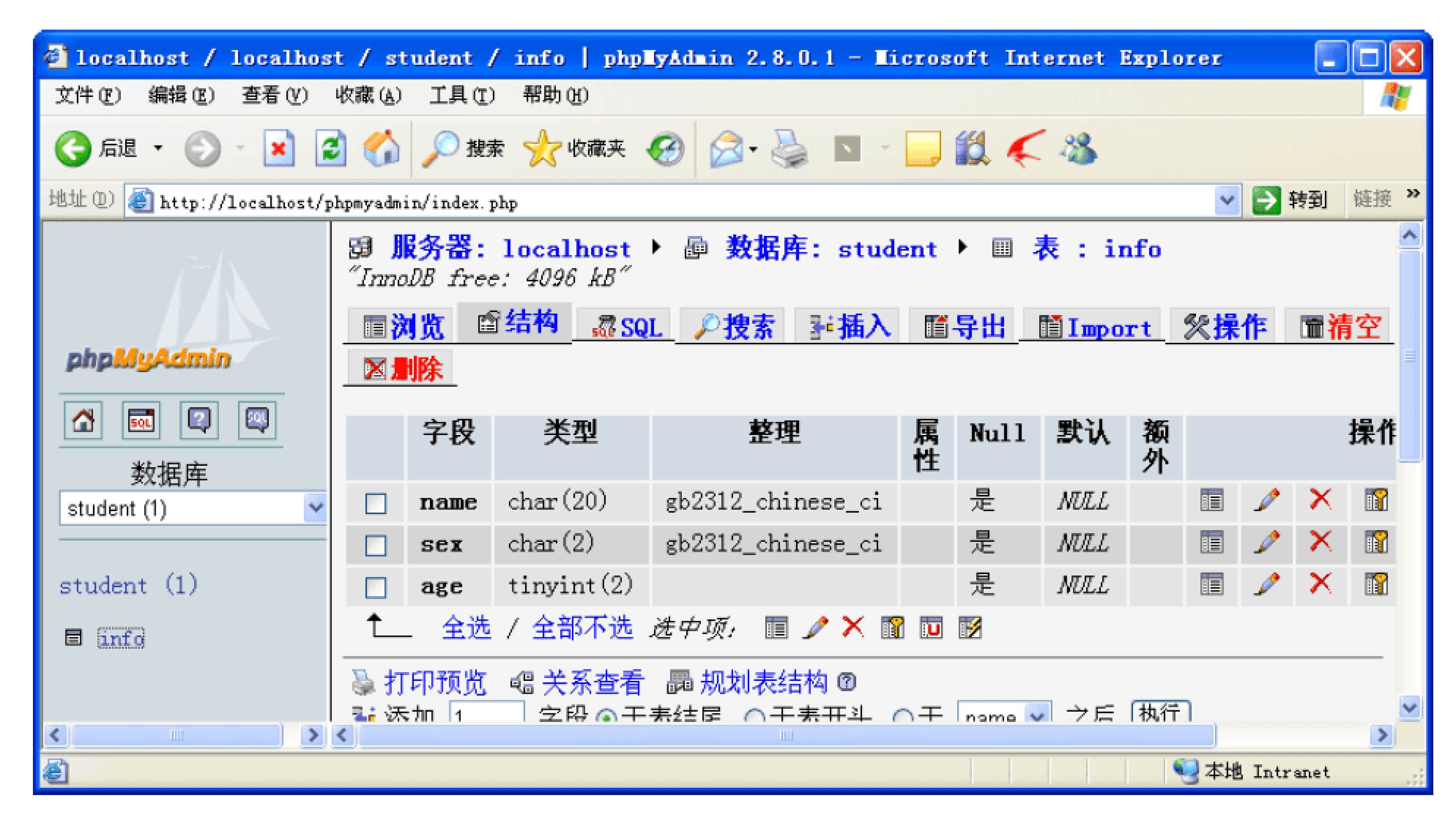


图 9-14　浏览表信息

这时单击右侧上方的按钮，可以进行浏览表内记录、表的结构、执行 SQL 语句、搜索、插入数据、导出数据等操作。

4. 浏览及编辑数据

单击表上方的“浏览”之后，会打开该表的记录。在此视图中可以对各条记录进行浏览、修改、删除等操作，如图 9-15 所示。



图 9-15　记录的浏览和编辑

phpMyAdmin 是一款功能强大的软件，可以很容易地对 MySQL 数据库进行各种管理。本书限于篇幅，就不再详细讲解其使用方法，只是将此工具介绍给广大读者，请各位读者自行安装、试用。掌握其常用功能，对于提高数据库管理效率将起到很大的帮助。

9.8 本章小结

本章主要介绍了 MySQL 数据库的安装和配置步骤、MySQL 的管理、SQL 的基本情况以及常用的 SQL 语句的写法。此外，还介绍了一款功能强大的 MySQL 数据库管理工具 phpMyAdmin。通过本章的学习，读者应熟练掌握 MySQL 数据库的安装配置方法，逐步掌握 MySQL 数据库的管理操作，不仅能够熟练使用基本的 SQL 语句进行数据库操作，还要会使用 phpMyAdmin 进行数据库管理。

9.9 练习题

1．MySQL 安装过程中有哪些值得注意的问题？

2．在 Web 开发中使用数据库有何优点？

3．常用的 SQL 语句有哪些？如何对查询结果进行排序？

4．MySQL 数据库的用户信息存储在什么位置？

5．在 MySQL 中新增一个用户有几种方法？分别如何操作？

6．怎样给 MySQL 用户授予不同权限？

7．上机课结束了，怎样将创建的数据库复制到 U 盘上？

8．用 phpMyAdmin 创建一个物流仓库数据库 hw，其中包括 3 张表：

（1）产品表。表中字段为产品编号、产品名称、价格和库存量。

（2）发货表。表中字段为销售日期、产品编号、客户编号、数量和金额。

（3）客户表。表中字段为客户编号、客户名称、负责人和联系电话。

第10章 PHP 操作 MySQL 数据库

本章重点

- PHP 操作 MySQL 数据库的流程;
- 常用 PHP 数据库函数;
- 数据的插入、删除和修改;
- PHP 数据分页的实现;
- MySQLi 扩展库的操作;
- PDO 操作 MySQL 数据库。

10.1 PHP 操作 MySQL 概述

从 PHP 5 开始，PHP 开发者放弃了对 MySQL 的默认支持，而是放到了扩展函数库中。因此，要使用 MySQL 函数，需要首先开启 MySQL 函数库。

打开 php.ini，找到“;extensions=php_mysql.dll”，将此行面前的分号“;”去掉，保存之后重新启动 IIS/Apache。运行第 5 章中的 show_info.php 文件，如图 10-1 所示。然后查找里面有没有一个名为 MySQL 的项目。如果能找到，则说明 PHP 已经完全开启了对 MySQL 的支持，可以在程序中直接调用 MySQL 数据库了。

如果此时 phpinfo()程序仍然显示不出 mysql 的信息，说明配置还没有成功。除了继续检查上一步修改是否正确以外，可以把 PHP 安装目录下的 libmysql.dll 这个库文件直接复制到系统的 system 目录或 system32 目录下。复制之后再重新启动 IIS/Apache，这时再次运行 show_info.php 程序，看是否出现了 mysql 信息。一般来说，复制 libmysql.dll 是最有把握的一种方法，正常情况下一定可以成功。

PHP 提供了大量函数，使用户可以方便地使用 PHP 连接到 MySQL 数据库，并对数据进行操作。学习 PHP + MySQL 数据库编程，首先要了解这些函数，明确具体的步骤，然后才能进入实质性开发阶段。

PHP 中可以用来操作 MySQL 数据库的函数如表 10-1 所示。

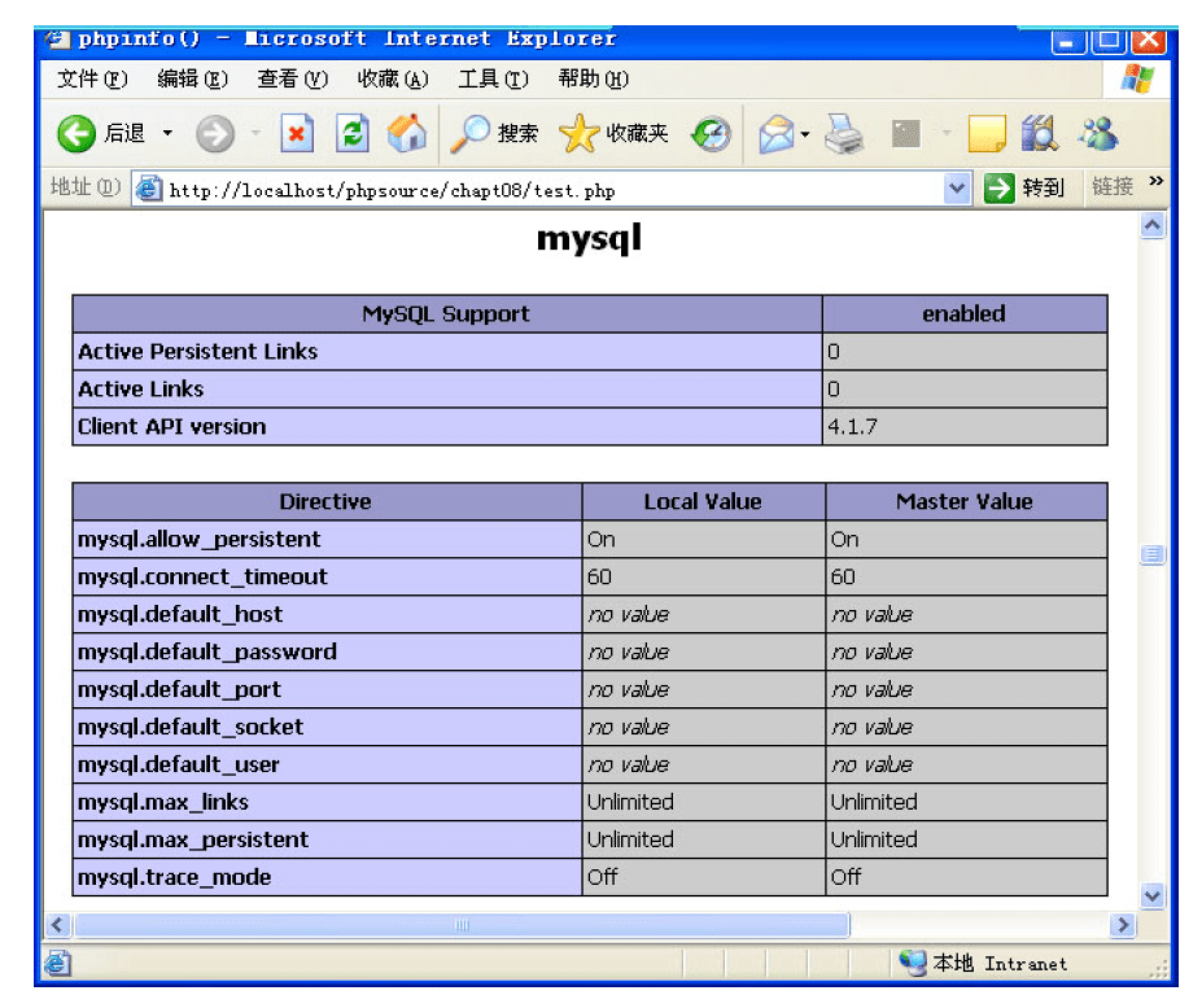


图 10-1　开启 MySQL 函数库

表 10-1　PHP 操作 MySQL 数据库的函数

函　数　名	功　　能
mysql_affected_rows	取得前一次 MySQL 操作所影响的记录行数
mysql_change_user	改变活动连接中登录的用户
mysql_client_encoding	返回字符集的名称
mysql_close	关闭 MySQL 连接
mysql_connect	打开一个到 MySQL 服务器的连接
mysql_create_db	新建一个 MySQL 数据库
mysql_data_seek	移动内部结果的指针
mysql_db_name	取得结果数据
mysql_db_query	发送一条 MySQL 查询
mysql_drop_db	丢弃（删除）一个 MySQL 数据库
mysql_errno	返回上一个 MySQL 操作中错误信息的数字编码
mysql_error	返回上一个 MySQL 操作产生的文本错误信息
mysql_fetch_array	从结果集中取得一行作为关联数组或数字数组，或两者兼有
mysql_fetch_assoc	从结果集中取得一行作为关联数组
mysql_fetch_field	从结果集中取得列信息并作为对象返回

续表

函　数　名	功　　能
mysql_fetch_lengths	取得结果集中每个输出的长度
mysql_fetch_object	从结果集中取得一行作为对象
mysql_fetch_row	从结果集中取得一行作为枚举数组
mysql_field_flags	从结果中取得和指定字段关联的标志
mysql_field_len	返回指定字段的长度
mysql_field_name	取得结果中指定字段的字段名
mysql_field_seek	将结果集中的指针设定为制定的字段偏移量
mysql_field_table	取得指定字段所在的表名
mysql_field_type	取得结果集中指定字段的类型
mysql_free_result	释放结果内存
mysql_get_client_info	取得 MySQL 客户端信息
mysql_get_host_info	取得 MySQL 主机信息
mysql_get_proto_info	取得 MySQL 协议信息
mysql_get_server_info	取得 MySQL 服务器信息
mysql_info	取得最近一条查询的信息
mysql_insert_id	取得上一步 INSERT 操作产生的 ID
mysql_list_dbs	列出 MySQL 服务器中所有的数据库
mysql_list_fields	列出 MySQL 结果中的字段
mysql_list_processes	列出 MySQL 进程
mysql_list_tables	列出 MySQL 数据库中的表
mysql_num_fields	取得结果集中字段的数目
mysql_num_rows	取得结果集中行的数目
mysql_pconnect	打开一个到 MySQL 服务器的持久连接
mysql_ping	ping 一个服务器连接，如果没有连接则重新连接
mysql_query	发送一条 MySQL 查询
mysql_result	取得结果数据
mysql_select_db	选择 MySQL 数据库
mysql_stat	取得当前系统状态
mysql_tablename	取得表名
mysql_thread_id	返回当前线程的 ID

10.2　PHP 对 MySQL 基本操作

图 10-2 所示为从 PHP 代码到最终取得数据所涉及的基本操作，这些操作都由对应的函数完成。

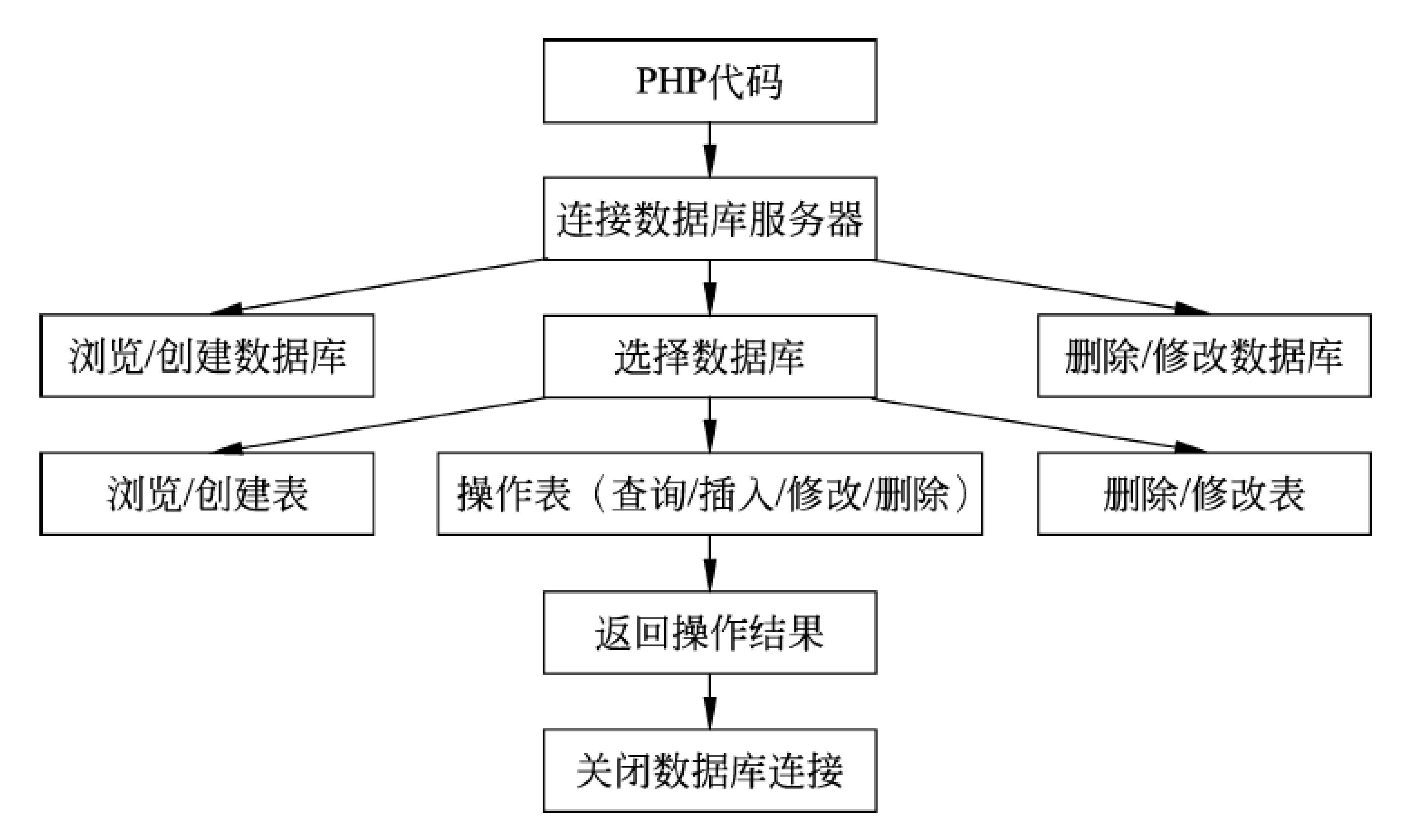


图 10-2　PHP 对 MySQL 的基本操作

在这些函数中，最常用的有 mysql_connect()、mysql_select_db()、mysql_query()、mysql_fetch_array()、mysql_num_rows()、mysql_close()等，下面就着重介绍这几个函数的使用。

10.2.1　在 PHP 中操作 MySQL

1. 建立连接

要在 PHP 中操作 MySQL 中的数据，第一步就是连接到 MySQL 数据库服务器，也就是建立一条 PHP 程序到 MySQL 数据库之间的通道。这样，PHP 才能通过这个通道向 MySQL 服务器发送各种指令，并取得指令执行的结果，将这些结果应用于 PHP 程序中。mysql_connect 函数就是用来建立和 MySQL 数据库连接的。

mysql_connect 函数有 5 个参数，但是通常情况下只用前 3 个参数，其格式如下：

```
resource mysql_connect (string server, string username, string password)
```

该函数返回类型为 resource 型，即资源型。3 个参数分别为 MySQL 服务器地址、MySQL 用户名、密码。这里的用户名可以用超级管理员的，也可以用用户表中存在的其他用户。如下面的语句将用超级管理员身份建立一个到本地服务器的连接：

```
$id=mysql_connect ("localhost","root","1234");
```

其中，localhost 换成 127.0.0.1 或本地机器的实际 IP 地址，效果都是相同的。另外，服务器地址后面可以指定 MySQL 服务的端口号，如果是采用默认的 3306 端口，则不必指定。如果采用了其他端口，则需要指定，如 127.0.0.1:88 表示 MySQL 服务于本地机器的 88 端口。用户名和密码均需指定（如密码为空，则直接用两个引号即可）。

将以上代码写在一个 PHP 程序中，写法如下：

```
<?php
     $id=mysql_connect("localhost","root","1234");
     echo $id;
?>
```

此程序运行之后，如果执行成功，则会输出一个资源型变量$id 的编号，类似于“Resource id #2”。如果执行失败，则有多种可能。如果出现下列提示：

```
Fatal error: Call to undefined function mysql_connect in […]
```

说明本地服务器的 MySQL 扩展库尚未被载入，因此 PHP 解释器无法识别 MySQL 函数。请参照 10.1 节的内容进行重新设置。

如果出现下列提示：

```
Warning: mysql_connect() [function.mysql-connect]: Unknown MySQL server host […]
```

说明 MySQL 服务器地址错误，可能是输入有错误，或服务器没有启动，或端口号不对。这时可以检查函数的第一个参数是否提供正确，MySQL 是否已成功启动。

还有可能出现下列提示：

```
Warning: mysql_connect() [function.mysql-connect]: Access denied for user […]
```

说明用户名或密码有错误，或本账号没有在 MySQL 服务器上登录的权限。

这里之所以如此详细地讲解该函数，就是因为这是连接到 MySQL 数据库的第一步。只要这一步成功了，下面的函数便都能运行。连接到数据库是一切工作的起点，因此必须保证此步骤成功，才能继续下面的内容。

2. 指定要操作的数据库

连接到数据库以后，还不能直接操作某个表，因为表都存储在各个数据库中，需要先选择要操作的数据库，才能对这个数据库中的表进行操作，mysql_select_db 函数就用来指定要操作的数据库。如果要操作的数据库还没有创建，则要先创建数据库，接着再创建数据库中的表。创建数据库和表使用 mysql_query 函数，参数是需要的 SQL 语句。程序 10-1 演示了用 PHP 程序在 MySQL 中创建一个数据库 jsj09，并在这个数据库中创建一个表 class1，表的字段为姓名、学号、年龄和籍贯。

程序 10-1.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-1.php: 用 PHP 创建新数据库和表-->
<head>
<title>10-1.php</title>
</head>
<body>
<?php
if (!$id=mysql_connect("localhost","root","1234")){
    echo "数据库服务器连接错误！";
    exit;         //如果数据库服务器连接不成功，退出程序执行
}
echo "数据库服务器连接成功！<br>";
if (!mysql_query("CREATE DATABASE jsj09",$id)){
    echo "数据库创建不成功，请检查账号权限和数据库是否已经存在！";
    exit;         //如果数据库创建不成功，退出程序执行
}
echo "数据库创建成功！<br>";
if (!mysql_select_db("jsj09",$id)){
    echo "数据库选择不成功！";
    exit;         //如果数据库选择不成功，退出程序执行
```

```
}
echo "数据库选择成功！<br>";
if (!mysql_query("CREATE TABLE class1(name varchar(10),id char(12) NOT
NULL PRIMARY KEY,age int(4),region varchar(10))",$id)){
    echo "数据表创建不成功，请检查 SQL 语句是否正确！";
     exit;        //如果 SQL 执行不成功，退出程序执行
}
echo "数据表创建成功！<br>";
if (mysql_close($id)){
     echo "数据服务器连接关闭成功！";
}
?>
</body>
</html>
```

程序执行的结果如图 10-3 所示。

图 10-3 程序 10-1.php 运行的结果

3. 向数据库中插入数据

使用 mysql_query 函数可以向数据库服务器发送任何合法的 SQL 指令（前提是数据库要支持该指令），参数是要执行的 SQL 语句。下面的程序 10-2 利用循环向服务器发送多次 INSERT 指令，向刚才创建的 class1 表中插入多条记录。

程序 10-2.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-2.php：用 PHP 向表中插入数据-->
<head>
<title>10-2.php</title>
</head>
<body>
<?php
$id=mysql_connect("localhost","root","1234");
mysql_select_db("jsj09",$id);
mysql_query("set name gb2312");
for($i=1;$i<6;$i++){
  $nl="145090".$i;
  $xm="张 0".$i;
  $sql="INSERT INTO class1 VALUES('$xm', '$nl',21, '山东')";
  $excu=mysql_query($sql,$id);
 if($excu){
       echo $sql;
       echo "第".$i."条数据插入成功！<br>";
  }else{
       echo "数据插入失败，错误信息：<br>";
       echo mysql_error();        //输出上一次 MySQL 执行的错误信息
  }
}
for($i=6;$i<10;$i++){
  $nl="145090".$i;
```

```
   $xm="刘 0".$i;
   $sql="INSERT INTO class1 VALUES('$xm', '$nl',21, '云南')";
   $excu=mysql_query($sql,$id);
  if($excu){
    echo $sql;
    echo "第".$i."条数据插入成功! <br>";
   }else{
    echo "数据插入失败, 错误信息: <br>";
    echo mysql_error();       //输出上一次 MySQL 执行的错误信息
   }
 }
 for($i=10;$i<20;$i++){
   $nl="14509".$i;
   $xm="李".$i;
   $sql="INSERT INTO class1 VALUES('$xm', '$nl',21, '安徽')";
   $excu=mysql_query($sql,$id);
  if($excu){
    echo $sql;
    echo "第".$i."条数据插入成功! <br>";
  }else{
    echo "数据插入失败, 错误信息: <br>";
    echo mysql_error();       //输出上一次 MySQL 执行的错误信息
        }
 }
 mysql_close($id);
 ?>
 </body>
 </html>
```

程序运行结果如图 10-4 所示。

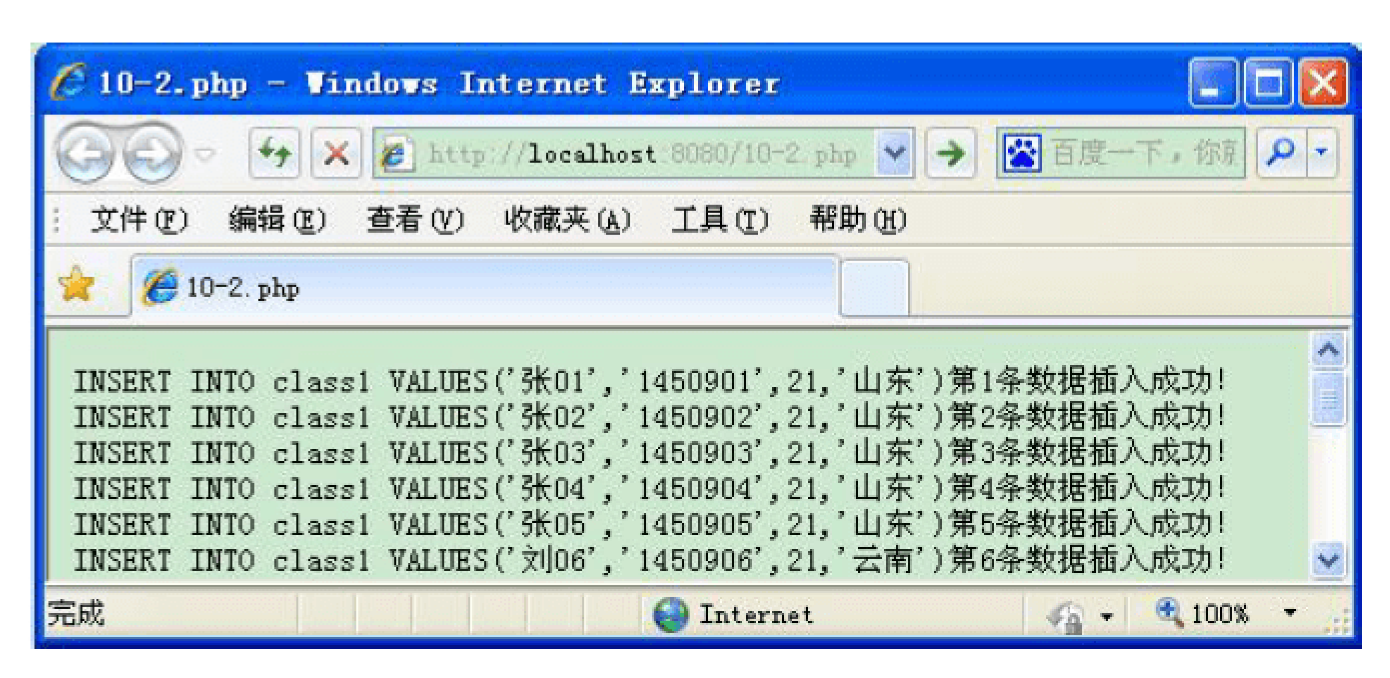


图 10-4　程序 10-2.php 的运行结果

4. 查询数据

一旦数据库中有了数据，就可随时查询数据。下面的程序 10-3 从数据库中读取上一个例子中插入的数据并用表格显示在网页上，用 mysql_query 函数，向数据库发送 SELECT 指令查询数据。

程序 10-3.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<!--程序 10-3.php：用 PHP 向表中插入数据-->
<head>
<title>10-3.php</title>
</head>
<body>
<?php
  $id=mysql_connect("localhost","root","1234");
  mysql_select_db("jsj09",$id);
  $query="SELECT * FROM class1";
  $result=mysql_query($query,$id);
  $datanum=mysql_num_rows($result);
  echo "表 class1 中共有".$datanum."条数据<br>";
?>
<table width="228" height="34" border="1">
  <?php  while ($info=mysql_fetch_array($result,MYSQL_ASSOC)){  ?>
  <tr>
    <td width="99" height="28"><?php echo $info["name"]?></td>
    <td width="113"><?php echo $info["id"]?></td>
    <td width="99" height="28"><?php echo $info["age"]?></td>
    <td width="113"><?php echo $info["region"]?></td>
  </tr>
<?php }?>
</table>
<?php mysql_close($id);?>
</body>
</html>
```

程序运行结果如图 10-5 所示。



图 10-5 程序 10-3.php 的运行结果

程序中用到了 mysql_fetch_array 函数。此函数是 PHP＋MySQL 编程中最常用的函数之一。此函数的格式如下：

```
array mysql_fetch_array (resource result [, int result_type])
```

该函数的作用是读取记录集 result 中的当前记录，将记录的各个字段的值存入一个数组中，并返回这个数组，然后将记录集指针移动到下一条记录。如果记录集已经到达末尾，

则返回 false。

第 2 个参数 result_type 为可选，此参数用来设置返回的数组采用什么样的下标。它有 3 个备选值：MySQL_ASSOC、MySQL_NUM、MySQL_BOTH，其含义如下：

① MySQL_ASSOC：返回的数组将以该记录的字段名称作为下标。如在本例中要输出此数组中的“姓名”字段，可以用$info[‘name’]。这里，$info 是数组名，name 是存放姓名的字段名。

② MySQL_NUM：返回的数组以从 0 开始的数字为下标。在本例中，返回的每条记录只有两个字段，那么数组也就只有两个元素，分别用$info[0]、$info[1]引用。

③ MySQL_BOTH：返回的数组既可以用字段名为下标，也可以用数字为下标。在本例中，既可以用$info[0]来取得姓名，也可以用$info[‘name’]来取得。

读者可以自行修改程序，对上述 3 个参数进行测试。

此外，PHP 中还有 mysql_fetch_row()、mysql_fetch_assoc()、mysql_fetch_object()等函数，这些函数的作用与用法都和 mysql_fetch_array 函数相似，读者可以参考 PHP 手册，了解这几个函数的使用。

用 mysql_query 函数，结合第 8 章讲过的 SQL 语句，还可以轻松实现对表内数据的删除和修改。例如：

```
<?php
    $id=mysql_connect("localhost","root","1234");
    mysql_select_db("jsj09",$id);
    mysql_query("DELETE FROM class1",$id);
    mysql_close($id);
?>
```

这段程序执行后，会删除表 class1 中的全部数据。

此外，用 UPDATE 语句可以实现对表内数据的修改，这里不再举例，读者可以自行编写程序练习。

5. 关闭连接

mysql_close 函数用来关闭一个数据库连接，其格式如下；

```
bool mysql_close ( [resource link_identifier] )
```

本函数只有一个可选参数 link_identifier。此参数表示要关闭的连接的 ID。也就是 mysql_connect 函数执行成功后返回的一个连接标记。参数为空时表示关闭当前连接。该函数返回一个布尔型结果。当关闭成功时返回 true，关闭失败是返回 false。

```
<?php
    $id=mysql_connect("localhost","root","1234");
    if(mysql_close($id)){
      echo"关闭数据库连接成功！";
    }else{
      echo"关闭数据库连接失败！";
    }
?>
```

上面的代码演示了 mysql_close 函数的使用。事实上，当一个 PHP 脚本（也是一个 PHP 页面以及它的包含文件）执行结束时，这个脚本中打开的 PHP 连接也会同时被关闭。因此

一般情况下即使忘记了手工关闭也没有关系。但是，数据库使用完毕后关闭连接是一个程序员应该有的编程习惯。

10.2.2 查询结果的分页显示

在 Web 开发中经常遇到的一个问题就是对大量数据进行分页显示。如一个留言板有数千条留言，如果这些留言全都显示在一个页面上，页面将变得很大，必须将数据分成一段一段的在页面上显示。PHP 中提供了非常简单的方法对数据进行分页。

程序 10-4.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-4.php：用 PHP 向表中插入数据-->
<head>
<title>10-4.php</title>
</head>
<body>
<?php
 $id=mysql_connect("localhost","root","1234")or die("连接服务器失败！");
  mysql_select_db("jsj09",$id)or die("选择数据库失败！");
  $query="SELECT * FROM class1";
  $result=mysql_query($query,$id)or die("查询数据失败！");
  $datanum=mysql_num_rows($result);
  if(isset($_GET ['page_id'])){
     $page_id=$_GET["page_id"];
  }else
  $page_id=1;
  $page_size="5"; //定义每页显示条数
  $page_num=ceil($datanum/$page_size);
?>
 表 class1 中共有<?php echo $datanum;?>条数据.
 每页<?php echo $page_size;?>条，共<?php echo $page_num;?>页。<br><br>
<?php
  for ($i=1;$i<=$page_num;$i++){
   echo "[<a href=?page_id=".$i.">".$i."</a>]";
  }
  $start=($page_id-1)*$page_size;
  $query2="SELECT * FROM class1 limit $start,$page_size";
  $result2=mysql_query($query2,$id)or die("查询数据失败！");
?>
<table width="228" height="34" border="1">
  <?php  while ($info=mysql_fetch_array($result2,MYSQL_ASSOC)){   ?>
  <tr>
  <td width="99" height="28"><?php echo $info["name"]?></td>
  <td width="113"><?php echo $info["id"]?></td>
  <td width="99" height="28"><?php echo $info["age"]?></td>
  <td width="113"><?php echo $info["region"]?></td>
  </tr>
```

```
40  <?php }?>
41  </table>
42  <?php mysql_close($id);?>
43  </body>
44  </html>
```

程序运行结果如图 10-6 所示。



图 10-6　程序 10-4.php 的运行结果

单击不同的页码，对应的数据也会随之变化，简单的分页功能实现了。程序说明如下：

（1）第 12～第 14 行，首先查询出了要显示的全部数据的条数，放在$datanum 中。

（2）第 15～第 19 行判断用户是否单击了某一页。通过第 15 行可以看到，当用户单击某一个页码时，将会把一个页码的数字作为参数传递给本页面（用 GET 方法在 URL 中传递参数），第 16～第 18 行就是判断是否传递了这一参数，如果传递了，则说明用户选择了某一页，这时当前要显示的页面就是传递过来的参数值。如果参数为空，则说明本页面是直接被打开，用户还没有选择某个页面。这时就显示第一页，因此设置$page=1。

（3）第 19 行定义了$page_size 变量，将其值设置为 5，即每页显示 5 条数据。

（4）第 20 行根据记录总数和每页显示的条数计算出了总页数，计算方法很简单，就不再分析了。这里，用到了 ceil 函数将计算结果进行进一取整。

（5）第 25～第 27 行，用循环输出了所有页码。每个页面都有一个超连接，单击此超连接会将相应的页码以 GET 方式传递到本页。

（6）第 28 行定义了变量$start，这个变量在第 29 行的 SQL 语句中被用到。MySQL 支持在 SELECT 语句中使用 LIMIT 子句，LIMIT 子句的使用方法如下：

```
SELECT * FROM tbl WHERE…LIMIT [begin],[num]
```

此查询语句的含义为，查询 tbl 表中满足指定条件的全部记录，然后只返回记录中从 begin 开始的 num 条数据。也就是说，使用了 LIMIT 子句后，MySQL 并不返回所有的满足条件的记录，而是只返回这些记录中从某条开始的若干条。如“LIMIT 10,5”的作用是取记录集中从第 10 条后的 5 条，即第 11、12、13、14、15 条记录。注意，这里的计数是从第 1 条开始的。

这样，第 28 行的$start 就不难理解了。记录截取起点的计算方法为：

（要显示的页码−1）× 每页显示的条数

如当前要显示第 3 页，每页 15 条，那截取起始位置就是(3−1) × 15=30。也就是说，第 3 页从第 31 条数据开始显示，连续显示 15 条，直到第 45 条。这与实际情况相符。

（7）第 33～第 38 行在一个表格中用 while 循环输出由第 29 行的 SQL 语句所查询的记录。

此程序虽然已经实现了分页功能，但是功能还不是很完善。如本例中全部页码直接用循环列出，这样虽然很方便，但是在页码很多（如几百页）的情况下会出现版面混乱。可以使用“上一页”、“下一页”、“首页”、“尾页”等方法，简化页码显示，甚至可以允许用

户输入跳转的页数。读者可以根据所学知识，完善此程序的功能。

10.3 面向对象的方式操作 MySQL

将前面操作 MySQL 的函数封装起来，构造一个操作 MySQL 的类，就可以用面向对象的方式操作 MySQL 了，这样在项目开发实践中有利于代码重用，提供开发效率。下面是一个具有操作 MySQL 功能的简单类，读者可以根据情况将其扩充，应用于自己的开发实践。

```
<?php
//文件 mysql.php,一个用来操作数据库的简单类
class mysql{
  private $link_id;
  private $db_host;
  private $db_user;
  private $user_psw;
  private $db_name ;
  private $charset ;
  public  function  __construct(  $db_host,$db_user,$user_psw,$db_name,
$charset='gb2312'){
    $this->db_host=$db_host;
    $this->db_user=$db_user;
    $this->user_psw=$user_psw;
    $this->db_name=$db_name;
  if(($this->link_id=mysql_connect($this->db_host,$this->db_user,
$this->user_psw )))
    {echo "Mysql 服务器连接成功! <br/>";}
    else{
      die("不能连接 MySQL 服务器: ".$this->db_host."<br/>");
    }
  mysql_query("set names ".$this->charset ,$this->link_id);
  if(mysql_select_db($this->db_name ,$this->link_id )===false){
     die("不能连接数据库: ".$this->db_name."<br/>");
     return false;
   }else
   { echo "数据库连接成功! <br/>";}
  }
public function reselect_database($db_name ){
  return mysql_select_db($this->db_name ,$this->link_id);
}
public function fetch_array($query,$result_type=MYSQL_ASSOC){
  return mysql_fetch_array($query,$result_type );
}
public function query($sql){
  return mysql_query($sql,$this->link_id);
}
public function affected_rows(){
  return mysql_affected_rows($this->link_id);
```

```
}
public function num_rows($query){
  return mysql_num_rows($query );
}
public function insert_id(){
  return mysql_insert_id($this->link_id );
}
public function selectByLimit($sql,$num,$start=0){
  if($start==0){
   $sql.= ' LIMIT '.$num;
  }else{
   $sql .= ' LIMIT '.$start . ', '.$num;
  }
  return $this->query($sql);
  }
public function getRow($sql ,$limted=false){
  if($limted ==true){
   $sql =trim($sql . 'LIMIT 1');
  }
  $result_type=$this->query($sql);
  if($result_type!==false){
    $row=mysql_fetch_row($res);
    return $row[0];
  }
  else{
    return false;
  }
}
public function getAll($sql ){
  $res=$this->query($sql );
  if($res!==false){
    $arr=array();
    while($row=mysql_fetch_assoc($res)){
      $arr[]=$row ;
    }
    return $arr;
  }
  else{
    return false;
  }
}
}
?>
```

将上述代码保存为 mysql.php，在需要的地方就可以使用了。程序 10-5.php 演示了用这个类操作前面创建的数据库 jsj09 的过程。

程序 10-5.php

```
<!--程序 10-5.php：MySQL 的面向对象操作方式-->
<?php
include("mysql.php");
```

```
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>10-5.php</title>
</head>
<body>
<?php
 $db=new mysql("localhost","root","1234","jsj09");
 $result=$db->selectByLimit("Select * from class1",7 );
if($result ){
  while($row=$db->fetch_array($result )){
    echo"姓名:".$row['name'].", 学号: ".$row['id'].
    ", 年龄: ".$row['age'].", 籍贯: ".$row ['region']."<br/>";
  }
}
?>
</body>
</html>
```

程序运行结果如图 10-7 所示。

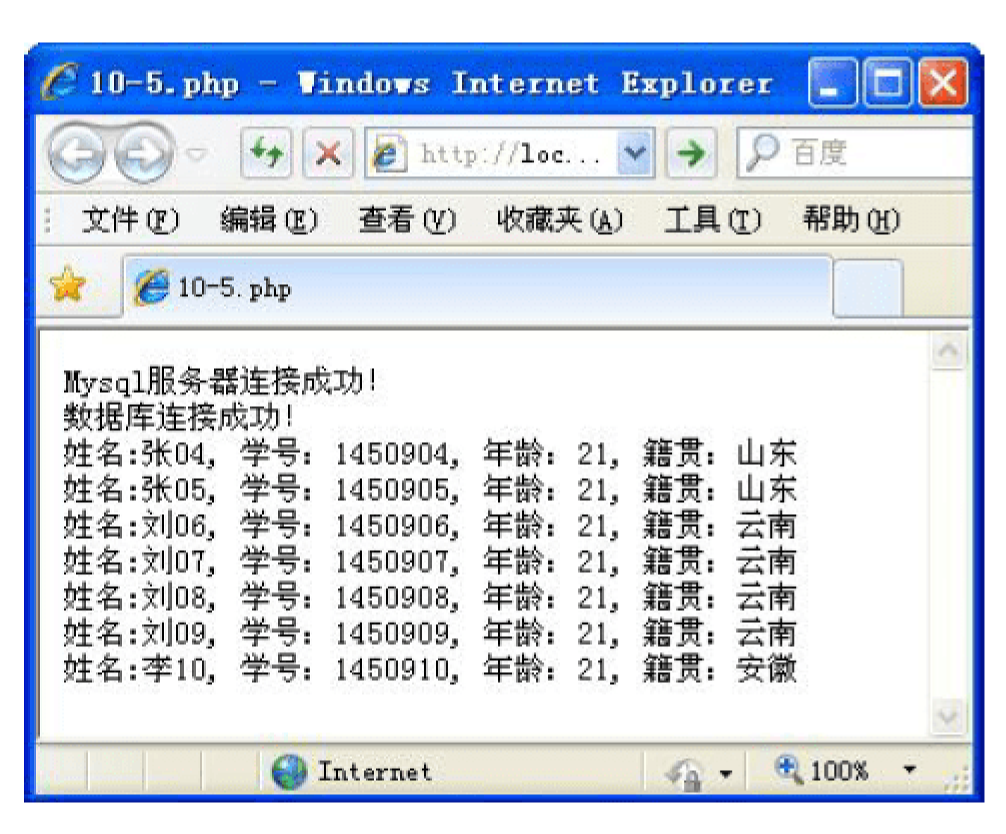


图 10-7　程序 10-5 运行结果

10.4　PHP 5 中的 MySQLi 扩展

人们都知道 PHP+MySQL 是构建应用系统的黄金搭配，其中 MySQL 扩展在两者之间提供了交互通道。MySQL 扩展提供的相关函数极大地方便了在 PHP 程序中存取 MySQL 数据库，但是，随着 MySQL 的发展，MySQL 扩展变得无法支持 MySQL4.1 及其更高版本的新特性。因此，人们又建立了一种全新支持 PHP 5 的 MySQL 扩展——MySQLi 扩展（i 表示改进），其执行速度更快。PHP 手册中说：如果使用的 MySQL 版本在 4.1.3 之后，强烈推荐使用 MySQLi 扩展。

在 PHP 5 以后的版本中，不仅可以使用早期的 MySQL 数据库操纵函数，而且还可以

使用 MySQLi 扩展技术实现与 MySQL 数据库的信息交流。利用 MySQLi 扩展技术不仅可以调用 MySQL 的存储过程、处理 MySQL 事务，而且还可以使访问数据库工作变得更加稳定。

MySQLi 扩展使得用户可以利用 MySQL 4.1 及其更高版本的新功能。与 MySQL 扩展相比，MySQLi 扩展在以下方面有了明显地提高：

（1）MySQLi 扩展可以很容易地使用 MySQL 的新功能，即使 MySQL 的新版本又出现了更多的功能，MySQLi 扩展也能很容易地支持。

（2）MySQLi 扩展不仅提供面向过程的编程方式，也可用面向对象的方式编程。MySQLi 扩展已经封装到一个类中。

（3）MySQLi 扩展执行的速度要比之前版本的 MySQL 扩展快了很多。MySQLi 扩展支持 MySQL 新版本的密码杂凑和验证程序，提高了应用程序的安全性。

（4）prepared 语句可提高重复使用的语句的性能，MySQLi 扩展提供了对预准本语句的支持。面向对象接口 prepared 表达式，有利于阻止 SQL 注入攻击。

（5）MySQLi 扩展进一步改进了调试功能，提高了开发效率。尤其重要的是 MySQLi 还能支持更多的表达式和事务处理。如果想支持多种数据库系统，还可以考虑 PDO。

要在 PHP 中使用 MySQLi 扩展，要在配置文件 php.ini 中添加如下设置：

```
extension=php_mysqli.dll
```

如果配置文件中已经有了上述设置，确保 extension 前面没有“;”号。

MySQLi 扩展中的库函数命名都是以 MySQLi 开始的，用法也和 MySQL 扩展中的函数类似。MySQLi 扩展库既可用面向过程的方式使用，也可以采用面向对象的方式使用。在 MySQLi 中，执行查询使用 query 方法，该方法的语法格式如下：

```
mixed query ( string $query [, int $resultmode ] )
```

其中，$query 参数为向服务器发送的 SQL 语句。resultmode 参数有两个取值，一个是 MYSQLI_STORE_RESULT，表示结果作为缓冲集合返回；另一个是 MYSQLI_USE_RESULT，表示结果作为非缓冲集合返回。

10.4.1　MySQLi 的面向过程方式用法

对前面的程序 10-4.php 修改，将程序中使用的 MySQL 扩展函数都换成 MySQLi 扩展的库函数，实现对数据库的操作。请读者注意这两个库中函数名和函数参数的变化。

程序 10-6.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-6.php：用 MySQLi 扩展访问数据库---->
<head>
<title>10-6.php</title>
</head>
<body>
<?php
$id=mysqli_connect("localhost","root","1234")or die("连接服务器失败!");
  mysqli_select_db($id,"jsj09")or die("选择数据库失败! ");
```

```
    $query="SELECT * FROM class1";
    $result=mysqli_query($id,$query)or die("查询数据失败！");
    $datanum=mysqli_num_rows($result);
    if(isset($_GET ['page_id'])){
       $page_id=$_GET["page_id"];
    }else{$page_id=1;}
    $page_size="5"; //定义每页显示条数
    $page_num=ceil($datanum/$page_size);
  ?>
   表 class1 中共有<?php echo $datanum;?>条数据.
   每页<?php echo $page_size;?>条，共<?php echo $page_num;?>页。<br><br>
  <?php
    for ($i=1;$i<=$page_num;$i++){
      echo "[<a href=?page_id=".$i.">".$i."</a>]";
    }
    $start=($page_id-1)*$page_size;
    $query2="SELECT * FROM class1 limit $start,$page_size";
    $result2=mysqli_query($id,$query2)or die("查询数据失败！");
  ?>
  <table width="228" height="34" border="1">
    <?php  while ($info=mysqli_fetch_array($result2,MYSQL_ASSOC)){   ?>
    <tr>
    <td width="99" height="28"><?php echo $info["name"]?></td>
    <td width="113"><?php echo $info["id"]?></td>
    <td width="99" height="28"><?php echo $info["age"]?></td>
    <td width="113"><?php echo $info["region"]?></td>
    </tr>
  <?php }?>
  </table>
  <?php mysqli_close($id);?>
  </body>
  </html>
```

程序运行结果如图 10-8 所示。



图 10-8 程序 10-6 运行结果

10.4.2　MySQLi 的面向对象方式用法

在面向对象的操作方式中，MySQLi 被封装成一个类，构造方法如下：

```
_ _construct([string $host[,string $username[,string $password[,string
$dbname[,int $port[,string $socket]]]]]])
```

其中，host 为连接的服务器地址。username 为连接数据库的用户名，默认值是服务器进程所有者的用户名。password 为连接数据库的密码，默认值为空。dbname 为连接数据库的名称。port 为 TCP 端口号。socket 为 UNIX 域 socket。下面的程序 10-7 将前面的程序 10-5 里的函数改成了 MySQLi 扩展中的函数并以面向对象的方式操作数据库。请读者注意两个程序中函数的不同用法。

程序 10-7.php

```
<!--程序 10-7.php: mysqli 的面向对象操作方式-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>10-7.php</title>
</head>
<body>
<?php
 $db=new mysqli("localhost","root","1234","jsj09");
 $query="Select * from class1 where region='云南'";
 $result=$db->query($query);
if($result ){
  while($row=$result->fetch_array()){
    echo"姓名:".$row['name'].", 学号: ".$row['id'].
    ", 年龄: ".$row['age'].", 籍贯: ".$row ['region']."<br/>";
  }
}
?>
</body>
</html>
```

程序运行结果如图 10-9 所示。



图 10-9　程序 10-7 运行结果

MySQLi 还提供了一个连接 MySQL 的成员方法 connect()。当实例化构造方法为空的 MySQLi 类时，用 MySQLi 对象调用 connect 方法同样可以连接 MySQL。用法如下：

```
$mysqli=new mysqli( );
$mysqli->connet($db_host,$db_user,$db_psw ,$db_name);
```

10.5 使用 PDO 访问数据库

假设，一家公司的系统是在 MS SQLServer 上运行 PHP，由于 MySQL 的快速发展引起了决策层的注意，公司需要将 MS SQLServer 更换成 MySQL。要移植代码，就得把所有和数据库有关的程序都改一遍。

PDO（PHP Data Objects）是一种数据库访问抽象层，就是多种数据库的一致接口。使用 PDO 可以在更换数据库时取得极大便利，极大提高程序运行效率，还可以通过预处理语句来防止 SQL 注入，具有更高的数据安全性。PDO 扩展为 PHP 访问数据库定义了一个轻量级的、一致性的接口，它提供了一个数据访问抽象层，这样，无论使用什么数据库，都可以通过一致的函数执行查询和获取数据。PDO 随 PHP 5.1 发行，旨在将常见的数据库功能作为基础提供，使用户能够轻松使用数据库。

在 Win32 环境中，由于 PHP 5.1 版本以上的压缩包里已经自带 PDO 扩展库文件，因此只要在 php.ini 文件中找到

```
;extension=php_pdo.dll
;extension=php_pdo_mysql.dll
```

将这两行前面的“；”号去掉，重启 IIS 即可。

PDO 中常用的函数如下。

1. PDO 连接数据库

PDO 提供了连接数据库的统一的接口——PDO 对象。

```
    $db=new PDO ("driver_name:dbname=db_name;host=hostname/IP; [charset=char_
type]", "db_username","db_password" );
```

说明：PDO 有 3 个参数。

（1）driver_name 是使用的 PDO 驱动，可以为 mysql, mssql, sybase, dblib, firebird, oci, odbc, pgsql, sqlite, sqlite2。

db_name 是数据库名称；hostname/IP 是指要连接到哪里，如果是本地则为 localhost。[charset=char_type]是可选的，用来设置字符类型。

（2）db_username 是用户名。

（3）db_password 是密码。

下面的代码连接到了前面使用的 jsj09 数据库：

```
<?php
$dsn='mysql: dbname=jsj09;host=localhost; ';
$user='root';
$password='1234';
try{
 if( $dbh=new PDO($dsn,$user,$password ))
  echo "kkkkkk" ;
}
catch(PDOException $e){
```

```
  echo 'Connection failed: '.$e->getMessage();
}
?>
```

2. PDO 获取数据

PDO 提供了很多与数据库交互的方法，如使用 query 方法。下面的程序 10-8.php 演示了用 PDO 扩展读取 jsj09 数据库的简单方法。

程序 10-8.php

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
02      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
03  <html xmlns="http://www.w3.org/1999/xhtml">
04  <!--程序 10-8.php: PDO 操作 MySQL-->
05  <head>
06  <title>10-8.php</title>
07  </head>
08  <body>
09  <?php
10  $dsn='mysql:dbname=jsj09;host=localhost; ';
11  $user='root';
12  $password='1234';
13  try{
14   $dbh=new PDO($dsn,$user,$password );
15  }
16  catch(PDOException $e){
17    echo 'Connection failed: '.$e->getMessage();
18  }
19  $sql="select * from class1 where region='山东'";
20  $dbh->quote($sql);
21  foreach($dbh->query($sql) as $row){
22    print $row['name']."\t";
23    print $row['id']."\t";
24    print $row['age']."\t";
25    print $row['region']."<br/>";
26  }
27  ?>
28  </body>
29  </html>
```

程序运行结果如图 10-10 所示。

图 10-10　程序 10-8 运行结果

10.6　MySQL 的高级访问

10.6.1　使用 prepare 语句

在对数据库进行查询时，query 方法无法自动地转义发送给它的任何数据，需要在执行 query 方法之前，用 quote 方法转义传递给 SQL 字符串的数据（如程序 10-8.php 第 18

行，quote 方法自动将字符型变量值的字符中首尾加上单引号，防止 SQL 注入，并且免去特殊符号转义的过程）。如果要重复执行许多次有不同参数但结构相同的查询，这个过程会占用大量的时间，这时应该使用 prepare 和 execute 方法，一个预处理语句，就可以避免重复分析、编译、优化的环节。简单来说，预处理语句使用更少的资源，执行速度也就更快。尤其是，这种方法能自动转义任何提供给它的参数，防止 SQL 注入攻击。下面的程序 10-9.php 演示了 prepare 语句的使用。

程序 10-9.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-9.php：使用 prepare 语句-->
<head>
<title>10-9.php</title>
</head>
<body>
<?php
$dsn='mysql:dbname=jsj09;host=localhost; ';
$user='root';
$password='1234';
try{
$dbh=new PDO($dsn,$user,$password );
$sql="INSERT INTO class1(name, id,age,region) VALUES (:name, :id,:age,
:region)";
$stmt=$dbh->prepare($sql);
$stmt->bindParam(':name', $name);
$stmt->bindParam(':id', $id);
$stmt->bindParam(':age', $age);
$stmt->bindParam(':region', $region);
$name="陈 03";
$id="1450921";
$age=23;
$region="内蒙";
$stmt->execute();
$name="陈 04";
$id="1450922";
$age=23;
$region="内蒙";
$stmt->execute();
$sql="select * from class1 where region='内蒙'";
$stmt=$dbh ->prepare($sql );
$stmt->bindParam(':name', $name);
$stmt->bindParam(':id', $id);
$stmt->bindParam(':age', $age);
$stmt->bindParam(':region', $region);
$stmt->execute();
while($row=$stmt->fetch(PDO::FETCH_ASSOC)){
  print $row['name']."\t";
  print $row['id']."\t";
```

```
  print $row['age']."\t";
  print $row['region']."<br/>";
 }
 }
 catch(PDOException $e){
  echo 'PDO Exception Caught. ';
  echo 'Error with the database:<br/>';
  echo 'SQL Query: '.$sql ;
  echo 'Error: '.$e->getMessage();
 }
 ?>
 </body>
 </html>
```

程序运行结果如图 10-11 所示。



图 10-11　程序 10-9 运行结果

10.6.2　错误处理

错误处理是软件开发中不可回避的问题。PDO 中有一个处理错误的类 PDOException，结构如下：

```
<?php
 class PDOException extends Exception
 {
    public $errorInfo = null;
                            // 错误信息，可用 PDO::errorInfo()或 PDOStatement::
                               errorInfo()访问
    protected $message;     // 异常信息，可用 Exception::getMessage()访问
    protected $code;        // SQL 状态错误代码，可用 Exception::getCode()访问
 }
 ?>
```

这个类继承自 PHP 中的 Exception 类，Exception 类的结构如下：

```
<?php
 class Exception
 {
    // 属性
    protected $message = 'Unknown exception'; // 异常信息
    protected $code = 0;                      // 用户自定义异常代码
    protected $file;                          // 发生异常的文件名
    protected $line;                          // 发生异常的代码行号

     // 方法
    final function getMessage();              // 返回异常信息
    final function getCode();                 // 返回异常代码
    final function getFile();                 // 返回发生异常的文件名
    final function getLine();                 // 返回发生异常的代码行号
    final function getTrace();                // backtrace() 数组
final function getTraceAsString();            // 已格式化成字符串的getTrace() 信息
 }
 ?>
```

在程序中可先用 PDO 类的 setAttribute 方法设置错误处理的模式，例如：

```
<?php
  try{
$dbh=new PDO($dsn,$user,$password);
 …
  $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}catch(PDOException $e){…}
?>
```

PDO 提供了 3 种不同的错误处理模式。

（1）PDO_ERRMODE_SILENT 是默认模式；只是使用 errorCode() 和 errorInfo() 方法设置要检查的错误代码，不显示错误信息。

```
 if (!$dbh->exec($sql)) {
 echo $dbh->errorCode() ."<BR>";
$info = $dbh->errorInfo();
// $info[0] == $dbh->errorCode() 统一的错误代码
// $info[1] 是驱动程序特定的错误代码
// $info[2] 是驱动程序特定的错误字符串
}
```

（2）PDO_ERRMODE_WARNING 除了设置错误代码之外，PDO 还会发出 PHP 警告，可以使用常规的 PHP 错误处理程序捕获该警告，或只是使该错误显示在浏览器中（在内部测试过程中非常有用）。

（3）PDO_ERRMODE_EXCEPTION 除了设置错误代码之外，PDO 还会抛出一个 PDOException，并将其属性设置为包含该错误代码和信息。然后，可以在代码的较高级别捕获该异常，使用全局异常处理程序捕获该异常，或不对其进行处理而终止脚本（如果异常导致脚本中断，事务处理回自动回滚）。

异常模式也是非常有用的，因为可以使用比以前那种使用传统的 PHP 风格的错误处理结构更清晰的结构处理错误，比使用默认模式使用更少的代码及嵌套，也能够更加明确地检查每个数据库访问的返回值。

在下面的程序 10-10.php 中，引入了一个错误，从表 class1 中访问一个不存在的字段，引发 PDO 异常。

程序 10-10.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-10.php：PDO 错误处理-->
<head>
<title>10-10.php</title>
</head>
<body>
<?php
$dsn='mysql:dbname=jsj09;host=localhost; ';
$user='root';
$password='1234';
try{
```

```
$dbh=new PDO($dsn,$user,$password );
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$sql="select * from class1 where asge=:age1";
$stmt=$dbh->prepare($sql);
$age =22;
$stmt->bindParam(':age1', $age,PDO::PARAM_STR);
$stmt->execute();
while($row=$stmt->fetch(PDO::FETCH_ASSOC)){
  print $row['name']."\t";
  print $row['id']."\t";
  print $row['age']."\t";
  print $row['region']."<br/>";
}
}catch(PDOException $e)
{echo 'Code: '.$e->errorInfo().'<br/>';
  echo 'PDO Exception Caught. ';
  echo 'Error with the database:<br/>';
  echo 'SQL Query:'.$sql ;echo'<pre>';
  echo 'Error:'.$e->getMessage().'<br/>';
  echo 'Code:'.$e->getCode().'<br/>';
  echo 'File:'.$e->getFile().'<br/>';
  echo 'Line:'.$e->getLine().'<br/>';
  echo 'Trace'.$getTraceAsString().'<br/>';
  echo '</pre>';
  }
?>
</body>
</html>
```

程序运行结果如图 10-12 所示。

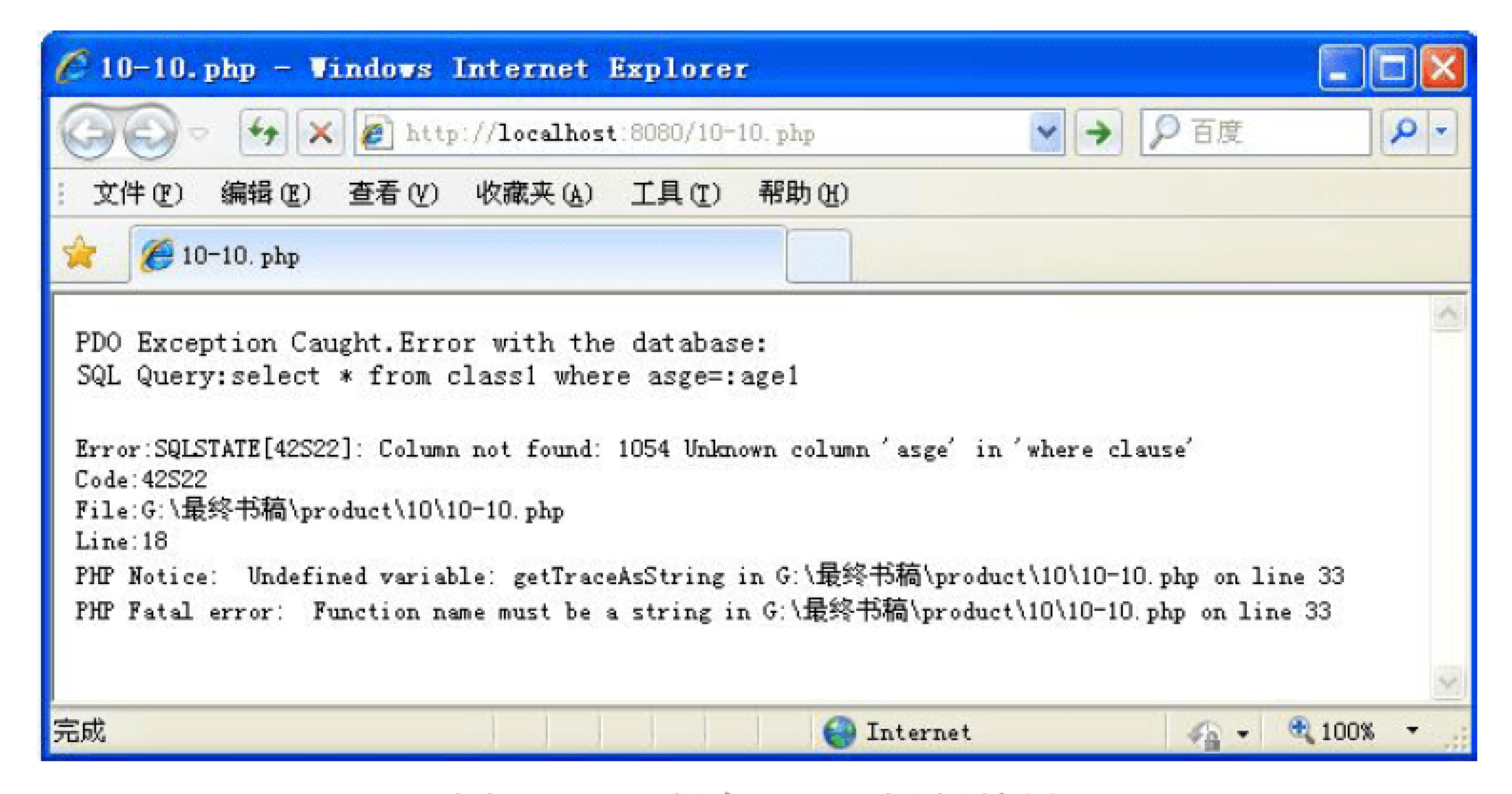


图 10-12　程序 10-10 运行结果

10.6.3　使用视图

有时需要的数据分布在几张表上，怎样用不同表上的数据构造一张新表，这就是视图解决的问题。视图是从一个或多个表或视图中导出的表，其结构和数据是建立在对表的查

询基础上的。和表一样，视图也是包括几个被定义的数据列和多个数据行，但就本质而言这些数据列和数据行来源于其所引用的表。所以，视图不是真实存在的基础表而是一张虚表，视图所对应的数据并不实际地以视图结构存储在数据库中，而是存储在视图所引用的表中。

视图一经定义便存储在数据库中，其中所含有的数据并不像真实表那样在数据库中再存储一份，通过视图看到的数据仍然是存放在基本表中的数据。对视图的操作与对表的操作一样，可以对其进行查询、修改(有一定的限制)、删除。

当对视图上的数据进行修改时，相应的基本表的数据也要发生变化。同时，若基本表的数据发生变化，则这种变化也可以自动地反映到视图中。使用视图，能够集中视点，简化操作，定制数据，并能增强数据的安全性。

为了更好地演示视图的使用，先在数据库 jsj09 中再创建一张 mark 表，该表有三列，一列是学号；两列是课程成绩。程序 10-11.php 创建表并插入数据，插入数据使用了 prepare 语句。

程序 10-11.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-11.php：使用视图（1）-->
<head>
<title>10-11.php</title>
</head>
<body>
<?php
$dsn='mysql:dbname=jsj09;host=localhost; ';
$user='root';
$password='1234';
try{
$dbh=new PDO($dsn,$user,$password );
$sql="create table mark
  (id char(8) not null primary key,
   cg int(6) not null,
   db int(6) not null
    )";
$stmt=$dbh->prepare($sql);
$stmt->execute();
$sql="INSERT INTO mark(id,cg,db) VALUES (:id,:cg,:db)";
$stmt=$dbh->prepare($sql);
$stmt->bindParam(':id', $id);
$stmt->bindParam(':cg', $cg);
$stmt->bindParam(':db', $db);
for($i=1;$i<6;$i++){
  $id="145090".$i;
  $cg=90+$i;
```

```
  $db=82+$i;
  $excu =$stmt->execute();
 }
 for($i=6;$i<10;$i++){
  $id="145090".$i;
  $cg=80+$i;
  $db=81+$i;
  $excu =$stmt->execute();
 }
 for($i=10;$i<20;$i++){
   $id="14509".$i;
   $cg=68+$i;
   $db=70+$i;
   $excu =$stmt->execute();
 }
 $sql="select * from mark";
 $stmt=$dbh->prepare($sql);
 $stmt->execute();
 while($row=$stmt->fetch(PDO::FETCH_ASSOC)){
   print $row['id']."\t";
   print $row['cg']."\t";
   print $row['db']."<br/>";
 }
 }
 catch(PDOException $e){
 echo 'PDO Exception Caught. ';
 echo 'Error with the database:<br/>';
 echo 'SQL Query: '.$sql ;
 echo 'Error: '.$e->getMessage();
 }
 ?>
 </body>
 </html>
```

程序运行结果如图 10-13 所示。

下面的程序 10-12.php 在表 class1 和 mark 的基础上构建视图 sumary，显示姓名、学号、图形学和数据库的成绩。

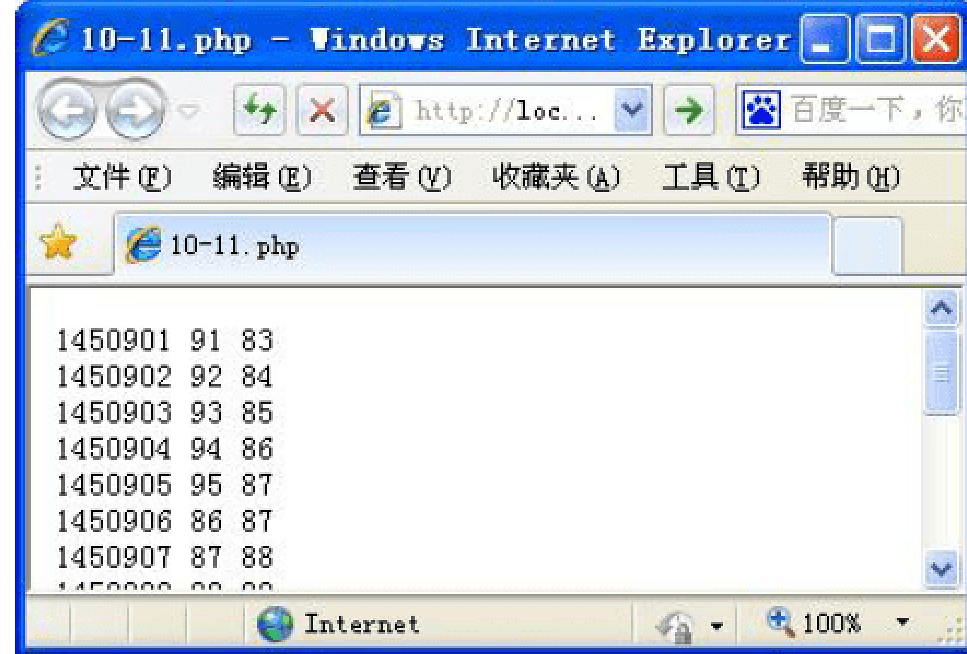


图 10-13　程序 10-11 运行结果

程序 10-12.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-12.php：使用视图（2）-->
<head>
<title>10-12.php</title>
</head>
```

```
<body>
<?php
$dsn='mysql:dbname=jsj09;host=localhost; ';
$user='root';
$password='1234';
try{
$dbh=new PDO($dsn,$user,$password );
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$sql="create view sumary as
 select class1.name as name,class1.id as id,mark.cg as cg,mark.db as db
  from class1,mark
  where class1.id=mark.id";
$stmt=$dbh->prepare($sql);
$stmt->execute();
$sql="select * from sumary";
$stmt=$dbh->prepare($sql);
$stmt->execute();
echo "   09 级计算机 1 班成绩表";
echo "<hr align=left size=1 width=150px color='blue'>";
echo "姓名"."   学号  "."   图形学"."   数据库"."<br/>";
while($row=$stmt->fetch(PDO::FETCH_ASSOC)){
  print $row['name']."\t"."\t";
  print $row['id']."\t"."\t";
  print $row['cg']."\t"."\t";
  print $row['db']."<br/>";
}
}
catch(PDOException $e){
echo 'PDO Exception Caught.';
echo 'Error with the database:<br/>';
echo 'SQL Query:'.$sql ;
echo 'Error:'.$e->getMessage();
}
?>
</body>
</html>
```

程序运行结果如图 10-14 所示。

图 10-14 程序 10-12.php 运行结果

10.6.4 事务处理

事务是对数据库的一个操作序列，这些操作要么全都执行，要么一个也不执行。事务的主要作用是在数据库发生错误或崩溃时确保数据库的一致性。MySQL 提供了市场上最强大的事务处理型数据库引擎，其功能包括完整的 ACID（atomic, consistent, isolated, durable）事务处理支持、无限的行级锁定、分布式的事务处理能力、读写互不冲突的多版本事务处理支持等。同时，通过服务器强制的参照完整性、专门的事务处理隔离

级别，以及即时的死锁检测来实现数据的完整性。PDO 中用 beginTransaction 方法开始一个事务时，用 commit 方法提交事务，用 rollBack 方法回滚。下面的程序 10-13.php 将表 class1 中 name 为“刘 02”的记录的 age 字段值改为 25，并插入一个 name 为“秦 11”，id 为 1450956,age 为 23，region 为“甘肃”的记录，这两个操作用事务的方式执行。

程序 10-13.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-13.php: PDO 事务处理-->
<head>
<title>10-12.php</title>
</head>
<body>
<?php
try {
$dsn = 'mysql:dbname=jsj09;host=localhost';
$user_name = 'root';
$user_psw = '1234';
$pdo = new PDO($dsn, $user_name, $user_psw);
$pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
$pdo->beginTransaction();
$sql1="update class1 set age=25 where name='刘 02'";
$sql2="insert into class1(name,id,age,region) values('秦 11','1450956',
23,'甘肃')";
$pdo->exec($sql1);
$pdo->exec($sql2);
$pdo->commit();
echo "事务执行成功。"."<br/>";
echo "请运行程序 10-3.php 查看执行后结果。"."<br/>";
}catch (Exception $e){
 $pdo->rollBack();
 echo $e->getMessage();
}
?>
</body>
</html>
```

程序运行结果如图 10-15 所示。运行程序 10-3.php 可观察表中数据的变化。

图 10-15　程序 10-13.php 运行的结果

10.6.5　存储过程

存储过程（Stored Procedure）是在大型数据库系统中的一组为了完成特定功能的 SQL 语句序列，经编译后存储在数据库中，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程能提高程序的速度，节省网络带宽；

通过对使用存储过程的用户授权，提高数据安全性；存储过程还可以重复使用，提高程序开发效率。MySQL 5.0 以后的版本开始支持存储过程，存储过程在项目开发中已得到广泛应用。

一个存储过程包括名字、参数列表，以及完成一定功能的 SQL 语句序列。MySQL 中存储过程用关键字 create procdure 开始，后面跟上存储过程名和参数。参数一般有 3 个部分：

（1）in、out 或 inout。

in 表示向存储过程传入参数，out 表示向外传出参数，inout 表示定义的参数可传入存储过程并可以被存储过程修改后传出存储过程。默认为 in。

（2）参数名。

（3）参数类型。

如果有多个参数，参数间用“，”号分隔。存储过程的 SQL 语句序列以 begin 开始，end 结束。SQL 语句中可以声明变量，使用控制语句，SQL 查询语句等。用关键字 drop 删除一个存储过程。下面的程序 10-14.php 使用 PDO 调用存储过程。

程序 10-14.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--程序 10-14.php：PDO 调用存储过程-->
<head>
<title>10-12.php</title>
</head>
<body>
<?php
try {
$dsn = 'mysql:dbname=jsj09;host=localhost';
$user_name = 'root';
$user_psw = '1234';
$pdo = new PDO($dsn, $user_name, $user_psw);
$pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
}catch (Exception $e){
  echo $e->getMessage();
}
$sql = "
create procedure myproce8(in par integer)
begin
declare varb varchar(25);
set varb='吉林';
insert into class1(name,id,age,region) values ('胡 4','33',20,varb);
select varb;
end;
";
```

```
$stmt=$pdo->prepare($sql);
$stmt->execute();
$sql = "call jsj09.myproce8(1);";
$stmt=$pdo->prepare($sql);
$stmt->execute();
$return_string=$stmt->fetch();
echo "存储过程执行成功。"."<br/>";
echo "请运行程序 10-3.php 查看执行后结果。"."<br/>";
echo $return_string [0];
?>
</body>
</html>
```

程序运行结果如图 10-16 所示。



图 10-16 程序 10-14.php 运行结果

10.7 本章小结

本章主要介绍了 PHP＋MySQL 编程的具体方法和步骤。首先，介绍 PHP 连接 MySQL 的原理和流程、连接前的注意事项；然后，介绍 MySQLi 的操作方法；最后，详细讲解了 PDO 操作 MySQL 数据库的方法。MySQL 数据库在 5.0 版本以后有了显著改进，建议读者尽量掌握 MySQLi 扩展库或 PDO 操作数据库的方法，熟练进行数据的插入、删除、修改和查询操作。

10.8 练习题

1. PHP 连接 MySQL 前需要做哪些准备工作？
2. 常用的 PHP 操作 MySQL 的函数有哪些？
3. 在 PHP 中如何对多条记录进行分页？
4. MySQL 中的函数与 MySQLi 中的函数在命名和功能上分别有什么不同？
5. 什么是 PDO？PDO 有何优点？
6. 编程实现一个分页显示的类。

7．编程实现用 MySQL 建立数据库连接的类。

8．用自己的所在班级的点名册作为数据表，分别用 MySQL、MySQLi 和 PDO 实现对数据的插入、查询和更新更能。

9．pepare 语句有什么优点？

10．什么是视图？实现视图的语句是什么？

11．什么是事务？MySQL 怎样实现事务处理？

12．什么是存储过程？存储过程有什么优点？

第 11 章 PHP 操作 XML 与 Ajax

本章重点

- XML 基本语法;
- PHP 解析 XML 文件的方法;
- Ajax 的概念;
- Ajax 处理数据的方法。

11.1 PHP 和 XML

第 2 章介绍了，XHTML 是学习网页设计的首选入门语言，XHTML 从出现到现在，标准在不断完善、功能也越来越强大，但是它的规范化要求依然不是很严格，仍有很多缺陷和不足。XHTML 有一个致命的缺点：只适合于人与计算机的交流，不适合计算机与计算机的交流。为此，W3C 开发了 XML 标准，可扩展标记语言（The Extensible Markup Language，XML）是 Web 设计的发展趋势，具有四大特点：优良的数据存储格式、可扩展性、高度结构化以及方便的网络传输。早期的 PHP 就支持 XML，PHP 4 增强了这种支持，而 PHP 5 则做得更好。本节将介绍 XML 的基础语法，并介绍如何用 PHP 5 的功能强大的 SimpleXML 创建和解析 XML 文件。

11.1.1 什么是 XML

XML 是一种使用一系列简单的标记描述数据的数据存储语言，它的主要目的是使用文本以结构化的方式来表示数据。XML 由互联网联盟（W3C）发布，它的简单使其易于在任何应用程序中读写数据，因此很快就发展成了数据交换的唯一公共语言。通过 XML，可以在不兼容的系统之间交换数据。这就意味着，程序可以更容易地与 Windows、Mac OS, Linux 以及其他平台下产生的信息结合，可以很容易地加载 XML 数据到程序中加以分析、处理，并以 XML 格式输出结果。

XML 虽然名称为可扩展标记语言，但它本身并不是一种标记语言。它提供了一个平台，能够制造自己的标记语言，这才是 XML 的可扩展的含义，也就是开发人员可以定义自己的一组标签，并使其他的人或程序能够理解这些标签。XHTML 是单标记语言，为特定应

用设计，而 XML 则是一系列的标记语言。因此，XML 比 XHTML 灵活得多。实际上，由于 XML 标签表示了数据的逻辑结构，不同的应用可以通过不同的方式来解释和使用这些标签。XML 使用标签对文档进行标记以提供有关内容的信息，不仅能加快搜索速度，而且还能降低网络流量。XML 允许开发人员定义任意数量的标签集，使开发人员有很大的灵活性决定要使用哪些数据，并确定数据的适用标准或自定义标签。对于 Internet 和大型企业 Intranet 环境，XML 能通过灵活、开放及基于标准的格式访问遗留数据库，将数据发送至 Web 客户端，提供了协同工作能力。这不仅可以更快地构建应用，而且更易于维护，还可以通过不同的样式表提供多个结构化数据的视图。同时，因为 XML 是个公共格式，不专属于任何一家公司，不依附于特定浏览器，不必担心被某些公司技术绑架。

11.1.2 XML 的基础语法

一个 XML 文档包括两个部分：XML 声明和数据。XML 声明必须位于文档的第一行，形式如下：

```
<?xml version="1.0" encoding="utf-8" ?>
```

XML 文档的主要部分是数据。这部分以一个根元素开始并结束，每一个 XML 文档都有一个唯一的根元素。在这个根元素之内的是一些嵌套的元素，每个元素包含一个起始标记、元素数据和对应的结束标记。下面是一个 XML 文档。

程序 11-1.php

```
00 <!--程序 11-1.xml：一个典型的 XML 文件-->
01 <?xml version="1.0" encoding="utf-8"?>
02 <jsj>
03  <textbook>
04   <title>PHP 动态网站开发教程</title>
05   <author>赵景秀</author>
06   <ISBN>978-7-302-16807-x</ISBN>
07   <press>清华大学出版社</press>
08   <date>2011-12</date>
09  </textbook>
10  <textbook>
11   <title>计算机图形学</title>
12   <author>Peter Shirley</author>
13   <ISBN>978-7-115-15867-3</ISBN>
14   <press>人民邮电出版社</press>
15   <date>2007-06</date>
16  </textbook>
17 </jsj>
```

其中，第 1 行是 XML 文档的声明，声明中指明了所遵守的 XML 是 1.0 版本。XML 有两个版本：1.0 和 1.1，目前使用 1.0 版本足够。声明中还指明所用编码为 utf-8 编码，如果用中文，需用 GB2312 编码。第 2 行为根元素<jsj>的开始，第 3～第 9 行是根元素的第一个元素<textbook>，这个元素中有<title>、<author>、<ISBN>、<press>和<date>五个嵌套的子元素。每个元素都有元素的开始标记、元素的数据和元素的结束标记。第 10～第 16 行

是根元素的第二个元素。第 17 行是根元素的结束标记。将这个文档保存为 jsj.xml。用浏览器查看结果如图 11-1 所示。

注意：XML 文档中的标记名称是大小写敏感的，名称中可以包含字母、数字和其他一些字符，但不能包含空格，开头不能用 XML，只能用字母或下划线。

XML 文档里可以加注释，如第 1 行：

```
<!--程序 11-1.xml:一个典型的 XML 文件-->
```

XML 的元素可以像 XHTML 的元素一样具有属性，属性必须有值，属性值必须放在引号中间，如第 2 行中属性 encoding 的值是 utf-8。

有些字符不能用于 XML 数据，这是要使用以“&”开头以“;”结束的字符组合，即预定义实体。预定义的 5 个实体如表 11-1 所示。

```
http://localhost:8080/11-1.xml - 原...
文件(F) 编辑(E) 格式(O)
1  <!--文件11-1.xml: 一个典型的XML文件-->
2  <?xml version="1.0" encoding="utf-8"?>
3   <jsj>
4    <textbook>
5     <title>PHP动态网站开发教程</title>
6     <author>赵景秀</author>
7     <ISBN>978-7-302-16807-x</ISBN>
8     <press>清华大学出版社</press>
9     <date>2011-12</date>
10   </textbook>
11   <textbook>
12    <title>计算机图形学</title>
13    <author>Peter Shirley</author>
14    <ISBN>978-7-115-15867-3</ISBN>
15    <press>人民邮电出版社</press>
16    <date>2007-06</date>
17   </textbook>
18  </jsj>
```

图 11-1 在 IE 浏览器查看 XML 文件

表 11-1 XML 的预定义实体

实 体	含 义
&	&
<	<
>	>
'	.
"	"

11.1.3 使用 SimpleXML 扩展

PHP 中操作 XML 文档的方法有多种，本章只介绍 SimpleXML 扩展。SimpleXMLS 是 PHP 家族中最后被加入的一个成员。从 PHP 5 开始，SimpleXML 的函数是 PHP 核心的组成部分，无须安装就可以使用。SimpleXMLS 处理 XML 文档功能强大，用法简单，尤其适合于处理类似记录的数据，是一种可靠、健壮的 XML 处理方法。表 11-2 所示为 SimpleXML 扩展中的函数。

表 11-2 SimpleXML 扩展提供的函数

函 数	描 述
__construct()	创建一个新的 SimpleXMLElement 对象
addAttribute()	给 SimpleXML 元素添加一个属性
addChild()	给 SimpleXML 元素添加一个子元素
asXML()	从 SimpleXML 元素获取 XML 字符串
attributes()	获取 SimpleXML 元素的属性

续表

函　　数	描　　述
children()	获取指定节点的子
getDocNamespaces()	获取 XML 文档的命名空间
getName()	获取 SimpleXML 元素的名称
getNamespaces()	从 XML 数据获取命名空间
registerXPathNamespace()	为下一次 XPath 查询创建命名空间语境
simplexml_import_dom()	从 DOM 节点获取 SimpleXMLElement 对象
simplexml_load_file()	从 XML 文档获取 SimpleXMLElement 对象
simplexml_load_string()	从 XM 字符串获取 SimpleXMLElement 对象
xpath()	对 XML 数据运行 XPath 查询

使用 SimpleXML 的第一步是用 simple_load_file 函数加载 XML 文档，将 XML 文档转换为一个 SimpleXML 对象，然后就可用一个 foreach 循环方便地处理对象中的每个元素。

下面的程序 11-2.php 创建 XML 文档，文档中记录了留言本中留言的序号、留言人、留言的内容、时间以及上传留言的机器的 IP 地址。

程序 11-2.php

```
<!--程序 11-2.php: 建立 XML 文件-->
<?php
if(trim(@$_POST['nickname'])==""or trim(@$_POST['content'])==""){
?>
<h4> 我的留言</h4><br/>
<div style=" border-style:solid; border-color: #CCC; border-width:1px;
width:400px; margin-left:20px">
 <form action="" method="post" name="guestbook" >
 <br/>
<textarea name="content" cols="60" rows="5"></textarea><br/>
 昵 称 : <input  name="nickname"  type="text"  maxlength="20"/><input
name="submit" type="submit" value="提交留言"/>
 </form>
 </div>
<p><a href="11-2.php">查看留言</a></p>
 <?php
}
else{
   header('Content-Type:application/xml;charset=utf-8');
   header('Cache-Control:no-cache,must-revalidate');
   header('Expires:Fri,14, Mar 2010 10:57:00 GMT');
   header('Last-Modified:'.date('r'));
   header('Pragram:no-cache');
   $dom=new DOMDocument('1.0', 'utf-8');
   if(file_exists("gb.xml")){
```

```
     $gb=simplexml_load_file(iconv('gb2312','utf-8','gb.xml'));
     foreach($gb->item as $item){
       $gbid_array[]=(int)$item ->id;
     }
     $gbid=max($gbid_array)+1;
     $gb=dom_import_simplexml($gb);
     $gb=$dom->importNode($gb,true);
    }
    else{
     $gb=$dom->createElement("guestbook");
     $gbid =1;
    }
    $item =$dom->appendChild($gb);
    $item=$dom->createElement("item");
     //id
    $id=$dom->createElement("id");
    $text=$dom->createTextNode($gbid );
    $id->appendChild($text);
    $id=$item->appendChild($id);
    //nickname
    $nickname=$dom->createElement("nickname");
    $abc=trim($_POST ['nickname']);
    $text=$dom->createTextNode( iconv('gb2312','utf-8',$abc));
    $nickname->appendChild($text);
    $nickname=$item->appendChild($nickname);
    //content
    $content=$dom->createElement("content");
     $abb=trim($_POST ['content']);
    $text=$dom->createTextNode( iconv('gb2312','utf-8',$abb));
    $content->appendChild($text);
    $content=$item->appendChild($content);
    //time
    $time=$dom->createElement("time");
    $text=$dom->createTextNode(date("Y-m-d H:i:s",time()));
    $time->appendChild($text);
    $time=$item->appendChild($time);
    //IP
    $IP=$dom->createElement("IP");
     $text=$dom->createTextNode($_SERVER['REMOTE_ADDR']);
     $IP->appendChild($text);
    $IP=$item->appendChild($IP);

    $item=$gb->appendChild($item);
    $dom->save("gb.xml");
    }
  ?>
```

程序创建的 XML 文件如图 11-2 所示，程序运行结果如图 11-3 所示。



图 11-2　程序 11-2.php 创建的 XML 文档

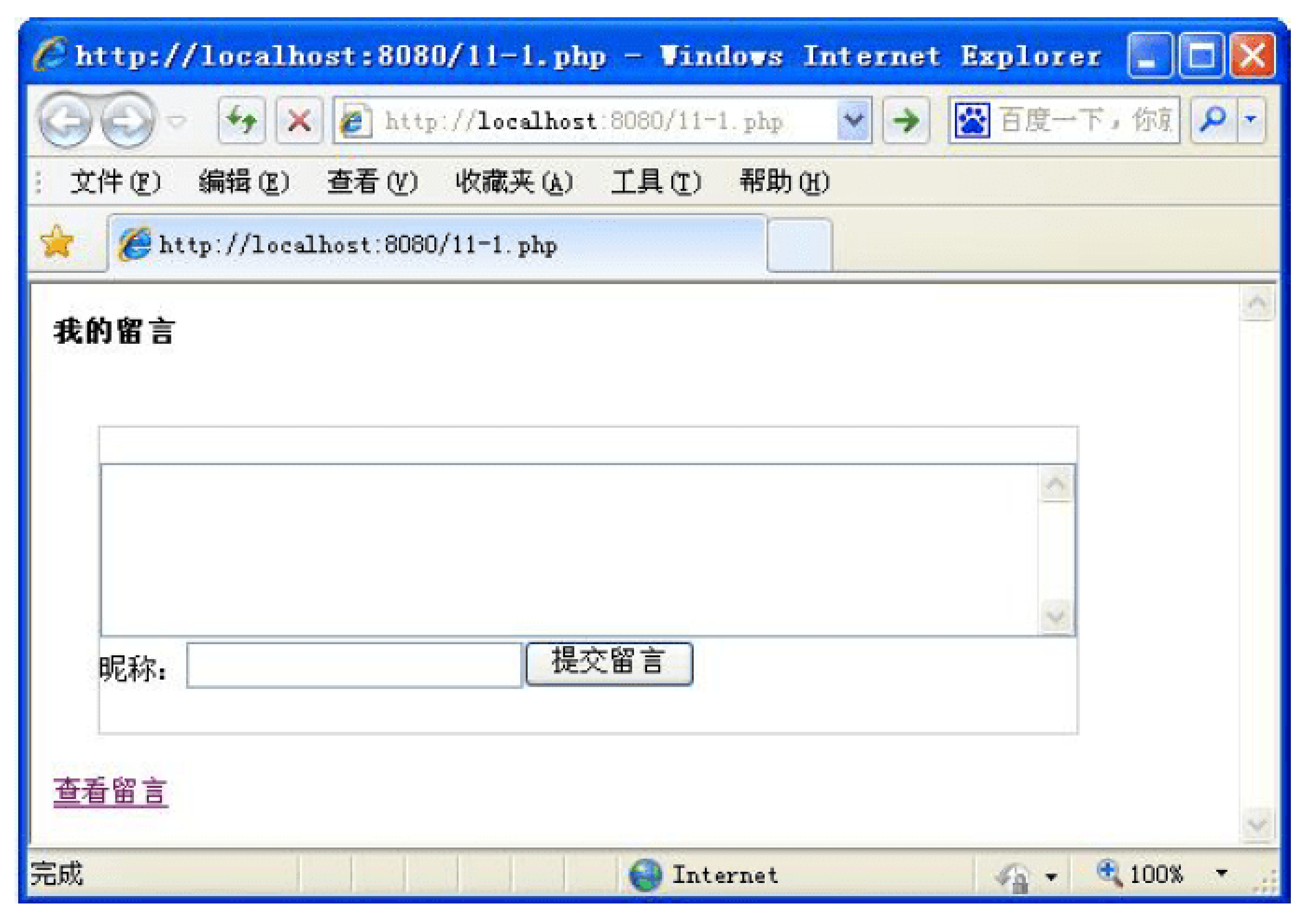


图 11-3　程序 11-2.php 运行结果

程序 11-3.php 则演示了读取程序 11-2.php 创建的 XML 文档并在页面上显示的方法。

程序 11-3.php

```
<!--程序 11-3.php：输出 XML 文件-->
<html>
<head>
  <title>11-3.php</title>
</head>
<body>
<?php
if(file_exists("gb.xml")){
```

```
//$gb=simplexml_load_file(iconv('UTF-8', 'gb2312', 'gb.xml'));
$gb=simplexml_load_file("gb.xml");
echo '<div style="width:410px;border-bottom-style:dashed; border-bottom-
color:#999;border-bottom-width:1px;">';
echo '</div>';
foreach($gb->item as $item){
echo '<div style=" background:#FFF; height:30px; width:450px; color:#36C;  ">';
echo "<br/>";
echo $item ->id."楼"." |";
echo "昵称:".iconv('utf-8','gb2312',$item ->nickname). " |";
echo "时间:".$item->time." |";
echo "来自:".$item ->IP."<br/>";
echo "</div><br>";
echo "留言:   ".iconv('utf-8',
'gb2312',$item->content)."<br/></br>";
echo '<div style="width:400px;border-bottom-style:dashed; border-bottom-color:
#999;border-bottom-width:1px;">';
 echo '</div>';
 }
}
else{
 echo "Gestbook XML file does not exist!";
}
?>
</body>
</html>
```

程序运行结果如图 11-4 所示。

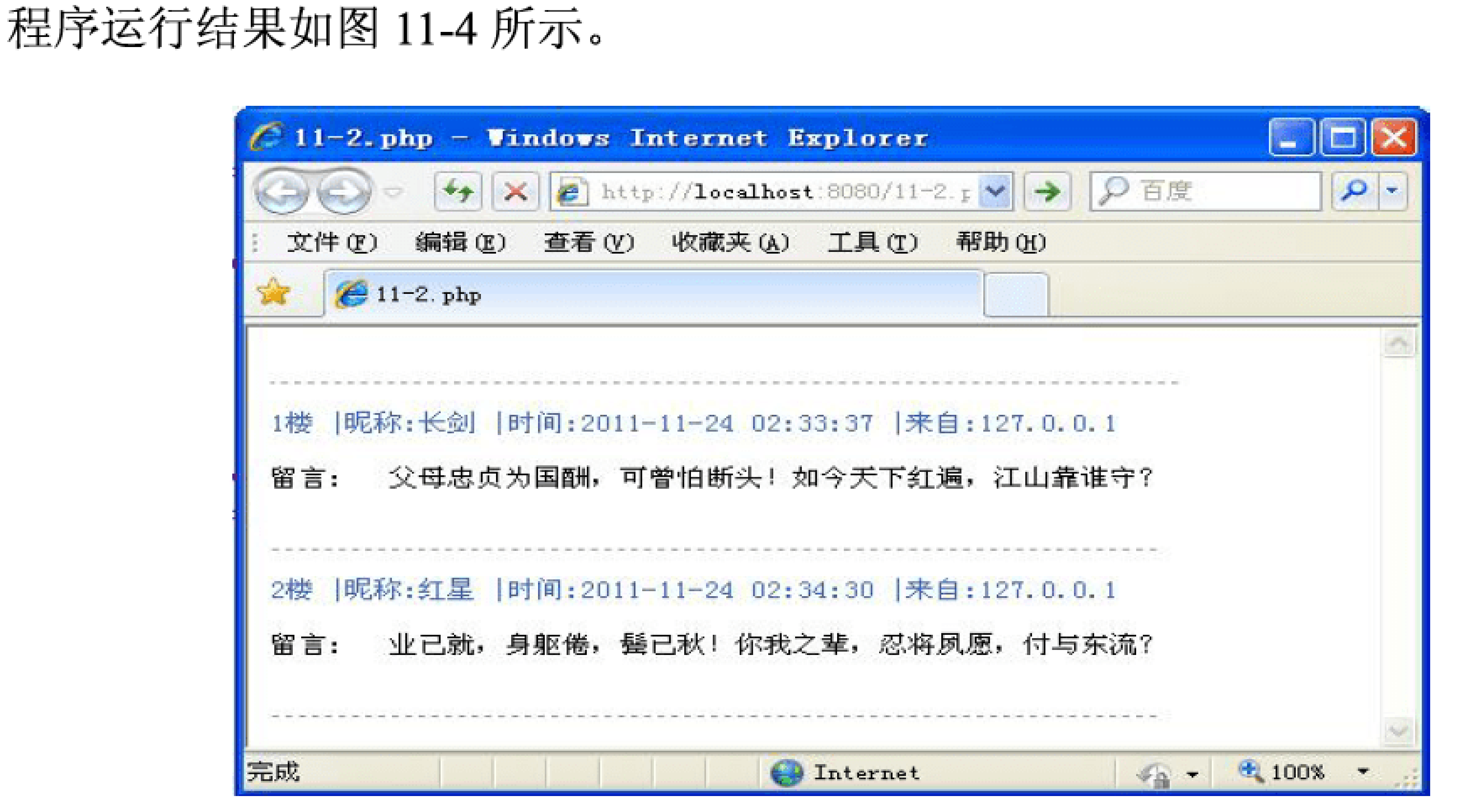


图 11-4　程序 11-3.php 运行结果

11.2　PHP 和 Ajax

Ajax（Asynchronous JavaScript And XML）是一种技术，把一部分原本属于服务器的工作转移到客户端，减轻服务器和带宽的负担，为基于 Web 的应用程序提供与桌面应用程

序接近的富用户界面和响应灵敏度，以达到更好的用户体验。2005 年以后的 Web 开发中大量使用了这种技术。

11.2.1 Ajax 是什么

为了更好地理解 Ajax，要从浏览器和服务器之间的通信说起。用户访问页面或提交表单，浏览器就会向服务器主机的某个端口请求一个 TCP 连接，服务器对连接请求进行确认，建立连接，接着浏览器发出请求页面报文，浏览器阻塞并等待服务器的回应。服务器处理请求并返回一个页面。浏览器接收返回的页面并重绘浏览器窗口，新的页面呈现给用户。这个过程中要花费一些时间，包括网络等待时间和新页面重绘时间。用户体验到的是，点击→等待→出现新页面→再点击→再等待→……。如果被传输的页面具有较多的数据量，用户就要等待，感觉到页面显示不顺畅。糟糕的情况下，会失去耐心，放弃请求。

如果用户请求的新文档只是修改或更新当前文档的相对较小的一部分内容，服务器也是要构造并发回完整的文档，整个浏览器窗口必须全部重新显示。Ajax 技术就是使网页从服务器请求少量的信息，使服务器只发回要更改或更新的那一小部分文档，即只提供发生变化的那部分。这样服务器缩短了传送文档时间，浏览器缩短了更新窗口的时间。这一切都发生在幕后，用户感到并没有离开当前的页面，但数据确实也更新了。这种想法虽然简单，但能更为迅捷地回应用户动作，大大改善 Web 用户的体验。

Ajax 的核心是 JavaScript 和文档对象模型。与传统 Web 应用不同，Ajax 采用的是异步交互的过程。异步的意思是通信过程中一方（如客户端）不必等待另一方（如服务端）就可以进行一些工作：用户进行一些操作，让 JavaScript 向服务器发出请求，在 JavaScript 等待数据的时候，用户还可以进行其他操作。具体来说就是，某一事件触发 Ajax 处理过程，浏览器先创建一个 XMLHttpRequest 对象，将 HTTP 方法（get/post）和目标 URL 以及请求返回后的回调函数设置到 XMLHttpRequest 对象上，通过 XMLHttpRequest 对象向服务器发送请求，发送请求后继续响应用户的界面交互，只有等待请求从服务器上返回时才调用函数，对响应进行处理。XMLHttpRequest 对象的常用方法如表 11-3 所示。

表 11-3 XMLHttpRequest 对象的常用方法

方　法	描　述
abort()	停止当前请求
getAllResponseHeaders()	把 HTTP 请求的所有响应的首部作为键/值返回
getResponseHeader(“header”)	返回指定首部的串值
open(String “method”, String “url”, bollean [asynch], String [username], String [password])	建立的服务器的调用，method 参数可以是 post、get 或 put。url 参数可以是相对 url 或绝对 url。这个方法还包括 3 个可选参数
send(content)	像服务器发送请求
setRequestHeader(“header”, “value”)	把指定首部设置为所提供的值。在设置任何首部之前必须现调用 open()

除了这些方法之外，XMLHttpRequest 对象还提供了一些属性，如表 11-4 所示。

表 11-4 XMLHttpRequest 对象的属性

属 性	描 述
onreadystatechange	每个状态改变时都会触发这个事件处理器，一般为调用一个 JavaScript 函数
readystate	请求的状态。有 5 个可取值，1=未初始化， 2=正在加载，3=已加载，4=交互中，5=完成
responseText	服务器的响应，表示为一个串
responseXML	服务器的响应，表示为 XML，这个对象一般解析为 DOM
status	服务器 HTTP 状态码（200 对应 OK，404 对应 not found 等）
statusText	HTTP 状态码的相应文本（OK 或 not found 等）

11.2.2 Ajax 处理数据的步骤

1. 初始化 Ajax

在使用 XMLHttpRequest 对象发送请求和处理响应之前，必须先用 JavaScript 创建 XMLHttpRequest 对象。XMLHttpRequest 不是 W3C 标准，不同的浏览器使用不同的方法。IE 浏览器将 XMLHttpRequest 实现为一个 ActiveX 对象，而其他浏览器将其实现为一个本地 JavaScript 对象。所以，创建该对象时，如果浏览器支持 ActiveX 对象，就使用 ActiveX 创建；否则就用本地 JavaScript 对象创建。创建 XMLHttpRequest 对象的代码如下：

```
<script>
function CreateXmlHttpObject ()
{
var xmlHttp=null;                   //定义一个变量
try {                               //Firefox, Opera 8.0+, Safari
 xmlHttp=new XMLHttpRequest();      //检查 XMLHttpRequest 对象是否可用
 }
catch (e){                          //IE
 try {                              //使用 Msxml2.XMLHTTP 组件创建对象，IE6 以后可用
 xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
  }
 catch (e){                         //用 Microsoft.XMLHTTP 组件创建 XMLHttpRequest 对象
  xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
 }
return xmlHttp;
}
</script>
```

2. 发送 HTTP 请求

一旦创建了 XMLHttpRequest 对象，就可以用其向服务器提交 HTTP 请求。方法是使

用 XMLHttpRequest 对象的 open 函数和 send 函数。首先用 open 方法建立对服务器的调用，然后再用 send 方法将请求发出。Open 方法的语法如下：

```
XMLHttpRequest.open("method", "url" [,flag]);
```

其中，参数 method 取值为 get 或 post,对应表单发送的 get 和 post 方法。参数 url 是页面要调用的相对或绝对地址。参数 flag 是个标记位，取值为 true 或 false，true 表示在等待被调用页面响应的时间内可以继续执行页面代码，false 则相反，默认为 true。

send 方法的语法格式如下：

```
XMLHttpRequest.send(content);
```

3. 指定响应处理函数

给服务器请求已经发出，当服务器返回信息时，需要指定客户端的处理方式，即编写一个处理函数并将函数名赋值给 XMLHttpRequest 对象的 onreadystatechange 属性。每当状态改变时都会触发这个事件处理器，通常会调用这个 JavaScript 函数：

```
xmlHttp.onreadystatechange=function_name
```

xmlHttp 为创建的 XMLHttpRequest 对象。函数名称 function_name 不加括号，不指定参数。也可以使用 JavaScript 即时定义函数的方法定义相应函数，例如：

```
XMLHttp.onreadystatechange=function()
{
      //代码
}
```

4. 处理服务器返回的信息

在进行操作前，处理函数首先需要判断请求的状态。表示请求状态的是 XMLHttpRequest 对象的 readyState 属性。通过判断该属性的值就可以知道请求的状态。有 5 个可取值，0 表示未初始化，1 表示正在加载，2 表示已加载，3 表示交互中，4 表示完成。

readyState 属性的值为 4 时，表示服务器已经传回了所有信息，可以开始处理信息并更新页面内容了。例如：

```
if(XMLHttp.readyState==4)
{
      //处理信息
}
else
{
      window.alert("请求还未成功");
}
```

服务器返回信息后需要判断服务器的 HTTP 状态码，确定返回的页面没有错误。通过判断 XMLHttpRequest 对象的 status 属性的值即可得到 HTTP 状态码。如 200 表示 OK（成功），404 表示 Not Found（未找到）。例如：

```
if(XMLHttp. status==200)
{
      //页面正常
}
```

```
else
{
    window.alert("页面有问题");
}
```

XMLHttpRequest 对象的 statusTextHTTP 属性保存了 HTTP 状态码的相应文本，如 OK 或 Not Found 等。

XMLHttpRequest 对成功返回的信息有两种处理方式：

（1）responseText。将传回的信息当字符串使用。

（2）responseXML。将传回的信息当 XML 文档使用，可以用 DOM 处理。

以下代码请求了服务器上的 try.txt 文件，并通过检测请求状态和返回码来确定是否输出文本中的文件内容。

```
<script type="text/javascript">
var xmlHttp=null;

try
 {
    // Firefox, Opera 8.0+, Safari
    xmlHttp=new XMLHttpRequest();
 }
catch (e)
 {
    // Internet Explorer
    try
    {
       xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (e)
    {
       xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
 }
return xmlHttp;
}

function SendRequest()
{
 CreateXMLHttpRequest();
 xmlHttp.onreadyStateChange=processRequest;
 xmlHttp.open("get","try.txt");
 xmlHttp.send(null);
}

function processRequest()
{
 if(xmlHttp.readState==4)
{
   if(xmlHttp.status==200)
{
```

```
      alert(xmlHttp.responseText);
    }
  }
 }
 </script>
```

请读者自行测试这段代码。

11.2.3 Ajax 应用举例

根据前面的介绍可知，Ajax 技术就是用 XMLHttpRequest 对象和服务器通信，将服务器返回的数据再用 DOM 对象处理。本节用两个例子演示 Ajax 在实际开发中的应用。

1. 读取 XML 文件

用 Ajax 读取 XML 文件，需要读取 XMLHttpRequest 对象返回的 responseXML 属性。

首先创建一个 XML 文件，这里用程序 11-1.xml。

```
<!--程序 11-1.xml：一个典型的 XML 文件-->
<?xml version="1.0" encoding="gb2312"?>
<jsj>
 <textbook>
  <title>PHP 动态网站开发教程</title>
  <author>赵景秀</author>
  <ISBN>978-7-302-16807-x</ISBN>
  <press>清华大学出版社</press>
  <date>2011-12</date>
 </textbook>
 <textbook>
  <title>计算机图形学</title>
  <author>Peter Shirley</author>
  <ISBN>978-7-115-15867-3</ISBN>
  <press>人民邮电出版社</press>
  <date>2007-06</date>
 </textbook>
</jsj>
```

然后再建立程序 11-4.php，代码如下：

程序 11-4.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<!--程序 11-4.php：Ajax 读取 XML 文件-->
 <title>11-4.php</title>
<script language="javascript">
  var xmlHttp;
  var browser="";
  function loadXML()
 {
    var fileRoute="11-1.xml"
```

```
    if (window.ActiveXObject)
    {
        xmlHttp = new ActiveXObject('Msxml2.DOMDocument');
        xmlHttp.async=false;
        xmlHttp.load(fileRoute);
        browser="ie";
    }
    else if (document.implementation && document.implementation.createDocument)
    {
     xmlHttp=document.implementation.createDocument('', '', null);
        xmlHttp.load(fileRoute);
        xmlHttp="other";
    }
    else
    {
    alert('未做与该浏览器的兼容！');
    }
}

  function getmessage()
  {
    var msg='';
    if(browser=="other")
    {
        var cNodes = xmlHttp.getElementsByTagName("book");
        for(j=0;j<cNodes.length;j++)
        {
                 var bookTitle=oNodes[j].childNodes[0].text;
                 var bookAuthor=oNodes[j].childNodes[1].text;
                 var ISBN=oNodes[j].childNodes[2].text;
                 var press=oNodes[j].childNodes[3].text;
                 var date=oNodes[j].childNodes[4].text;
 msg+='教程名称：'+bookTitle+'<br/>'+'教材作者：'+bookAuthor+
 '<br/>'+'书号：'+ISBN+'<br/>'+'出版社：'+press+'<br/>'+'出版时间：'
 +date+'<br/>';
        }
    }
    else if(browser=="ie")
    {
        var state = xmlHttp.readyState;
        if (state == 4)
        {
            var oNodes = xmlHttp.selectNodes("//jsj/textbook");
            for(j=0;j<oNodes.length;j++)
            {
               var bookTitle=oNodes[j].childNodes[0].text;
                var bookAuthor=oNodes[j].childNodes[1].text;
                 var ISBN=oNodes[j].childNodes[2].text;
                 var press=oNodes[j].childNodes[3].text;
```

```
                    var date=oNodes[j].childNodes[4].text;
 msg+='教程名称：'+bookTitle+'<br/>'+'教材作者：'+bookAuthor+
 '<br/>'+'书号：'+ISBN+'<br/>'+'出版社：'+press+'<br/>'+'出版时间：'
 +date+'<br/><br/>';
              }
          }
      }
    document.getElementById("textbooks").innerHTML=msg;
 }
   </script>
</head>
   <body onload="loadXML();">
<div id="textbooks" style="width:500px;"></div>
 <input name="button" type="button" onclick="getmessage()" value="选用教材" />
   </body>
   </html>
```

程序运行结果如图 11-5 所示。

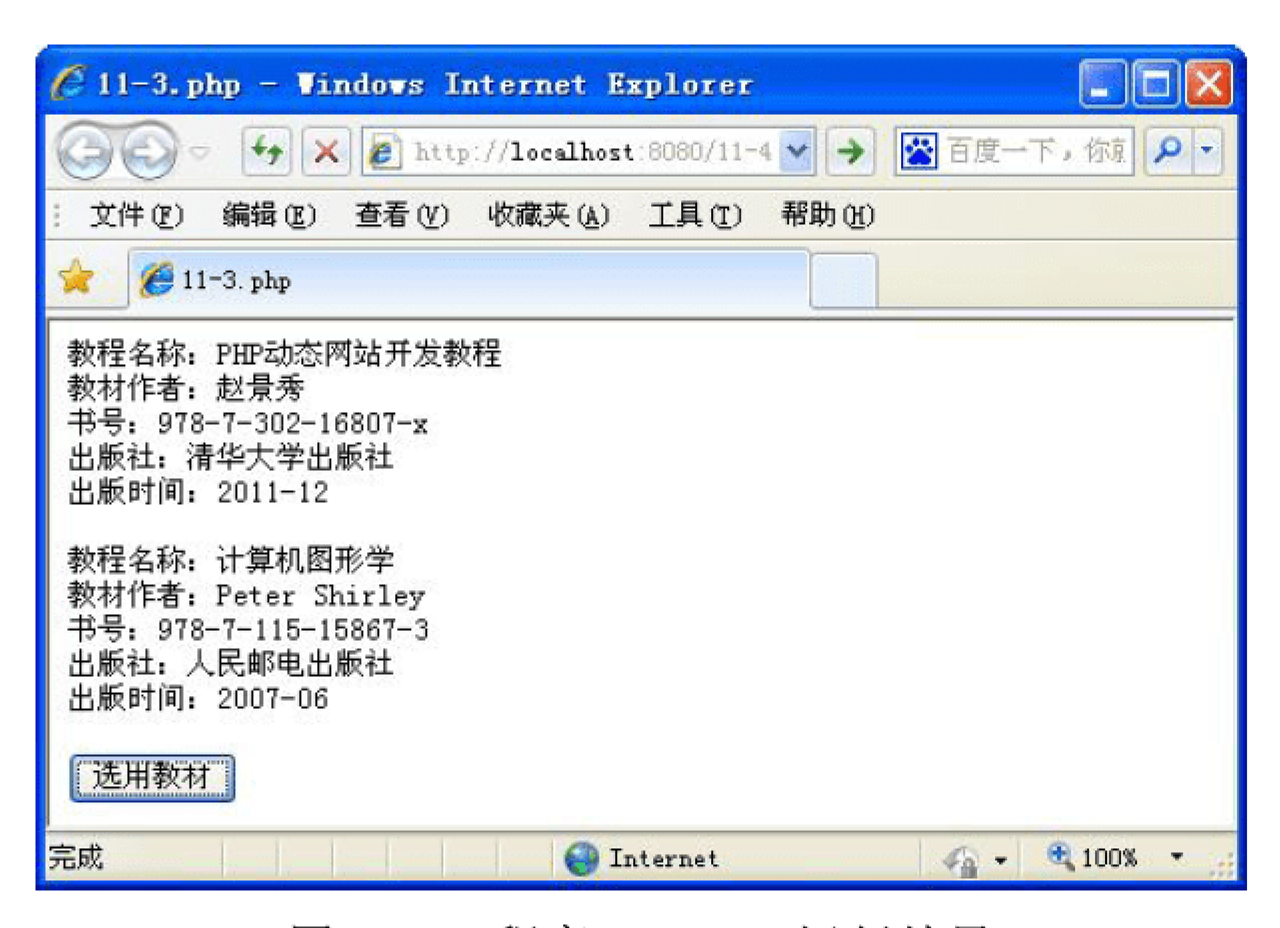


图 11-5　程序 11-4.php 运行结果

2. 读取数据库文件

注册表单程序是 Ajax 的一个典型应用。如果注册时用户选择的用户名已经被注册，就要告诉用户重新选择一个。在传统的注册方法中，用户要把所有表单上要填的项目填完提交后才检查用户的用户名是否可用。使用 Ajax 技术，可以在用户选择完用户名后及时验证该用户名是否可用，不必等到表单提交时，从而极大地方便了用户。

下面的例子程序 11-5.php 和程序 11-6.php 实现了这样的功能。用户在程序 11-5.php 提供的页面上注册，程序 11-6.php 对用户名进行后台验证。为了完成这个例子，需要先在数据库 jsj09 里创建一张表 users，表中共有 5 列：user_id、username、userpass、qq_number、nickname 和 email，并插入 3 行数据。第一行的用户名是 zhangkexin，用它来测试程序中 Ajax 的功能。

程序 11-5.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<!--程序 11-5.php：Ajax 验证注册页面-->
<head>
    <title>11-5.php</title>
<script language="javascript">
   var xmlHttp = null;
if (window.XMLHttpRequest) {
   xmlHttp = new XMLHttpRequest();
} else if (window.ActiveXObject) {
   try {
      xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
        try {
            xmlHttp= new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {alert("IE 浏览器版本需要升级！"); }
    }
}

function check_username(username) {
    if (xmlHttp) {
    xmlHttp.open('get', '11-5.php?username=' + encodeURIComponent(username));
        xmlHttp.onreadystatechange = pro_check;
        xmlHttp.send(null);
    }
}

function pro_check() {
if ( (xmlHttp.readyState == 4) && (xmlHttp.status == 200) ) {
  document.getElementById('username_label').innerHTML = xmlHttp.responseText;
    }
}
</script></head>
<body>
<form action="register.php" method="post">
<fieldset>
<legend>快速注册</legend>
 <p>用户名 : <input name="username" type="text" size="20" maxlength="20"
onchange="check_username(this.form.username.value)" />
<span id="username_label"></span></p>
<p>密 码: <input name="pass1" type="password" /></p>
<p>确认密码: <input name="pass2" type="password" /></p>
<p>QQ 号码: <input name="qq_number" type="text" size="20" maxlength="20" /></p>
 <p>昵   称: <input name="nickname" type="text" size="20"
maxlength="20" /></p>
  <p>电子邮箱: <input name="email" type="text" size="20" maxlength="60" /></p>
<input name="submit" type="submit" value="注册" />
</fieldset>
</form></body>
</html>
```

程序运行结果如图 11-6 所示。

图 11-6　程序 11-5.php 运行结果

在后台进行 Ajax 验证的程序如下：

程序 11-6.php

```
<!--程序 11-6.php: Ajax 后台验证程序-->
<?php
// 验证 11-5.php 传来的用户名是否可用
if (isset($_GET['username'])) {
  $db= @mysqli_connect ('localhost', 'root', '1234', 'jsj09') OR die ();
  $qq = sprintf("select user_id from users where username='%s'",
  mysqli_real_escape_string($db, trim($_GET['username'])));
  $result = mysqli_query($db, $qq);
   if (mysqli_num_rows($result) == 1) {
        echo 'Please enter another username!';
     } else {
        echo 'The username is OK!';
     }
     mysqli_close($db);
} else { // 处理没有输入用户名的情况
     echo 'Please enter a username.';
}
?>
```

程序运行结果如图 11-7 所示。

图 11-7　程序 11-6.php 运行结果

11.3　本章小结

本章着重介绍了 XML 和 Ajax 的概念，介绍了 XML 的基本语法和 Ajax 处理数据的步骤，分别列举了 PHP 读取并解析 XML 文档以及在 PHP 程序中使用 Ajax 进行异步通信的实用方法。XML 是未来的数据表示格式，在跨平台交换数据方面不可缺少。Ajax 的核心是用 XMLHttpRequest 对象和服务器通信，传输的数据量小，可在不刷新页面的同时更新数据，给用户带来了更好的心理感受。这两种技术在 PHP 开发中占有重要地位，希望读者通过本章的学习，能够在今后的实际开发中举一反三，熟练应用，提高自己的技术水平。

11.4　练习题

1．什么是 XML？其可扩展性主要是指什么方面？
2．XML 和 XHTML 在语法上的主要区别有哪些？
3．SimpleXMLS 扩展中的主要方法有哪些？各有什么功能？
4．什么是 Ajax？什么是异步通信？
5．Ajax 是怎样完成和服务器的通信的？
6．XMLHttpRequest 对象有哪些主要方法？各有什么功能？
7．XMLHttpRequest 对象的属性有哪些？
8．创建 XMLHttpRequest 对象时要注意什么？
9．用 XML 取代小型数据库，编程实现对数据的插入、查询和更新操作。
10．用 Ajax 编程实现级联下拉菜单。

参 考 文 献

1 曹衍龙，赵斯思．PHP 网络编程技术与实例．北京：人民邮电出版社，2006．

2 王黎，于永军，张豪．PHP+Dreamweaver CS4+CSS+Ajax 动态网站开发典型案例．北京：清华大学出版社，2010．

3 Davey Shafik，Matthew Weier O’phinney，Ben Balbo．PHP 深度分析．周广辉，杨建军，王春学译．北京：中国水利水电出版社，2010．

4 丁月光，孙更新，闫吉辉．PHP+MySQL 动态网站开发．北京：清华大学出版社，2008．

5 郑阿奇．PHP 实用教程．北京：电子工业出版社，2009．

6 邹天思，潘凯华，刘中华．PHP 数据库系统开发完全手册．北京：人民邮电出版社，2007．

7 梁胜民，肖新峰，王占中．CSS+XHTML+JavaScript 完全学习手册，北京：清华大学出版社，2008．

8 胡孟杰，郑延斌，岳明．JavaScript 动态网页开发案例指导．北京：电子工业出版社 2009．

9 唐四新．基于 Web 标准的网页设计与制作．北京：清华大学出版社，2009．

10 Larry Uilman．PHP 5 高级应用开发实践．王军，龚涛译．北京：人民邮电出版社，2008．

11 张兵义，吴燕军，袁彩虹．网站规划与网页设计．北京：电子工业出版社，2009．